FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

MasterVoicing - A whispers to voiced speech assistant

Maria João Araújo Soutelo

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof. Dr. Eng. Aníbal João de Sousa Ferreira

September 21, 2017

MasterVoicing - A whispers to voiced speech assistant

Maria João Araújo Soutelo

Mestrado Integrado em Engenharia Informática e Computação

September 21, 2017

Abstract

Aphonia, also known as loss of voice, is a condition that affects the human phonetic system and is characterized by the inability of a speaker to produce normal speech. It can range from partial loss, known as hoarseness, to an almost complete loss of voice, where the voice is nothing but a whisper. Its causes can vary, from physical disease related with injuries, medical procedures or bad habits, such as voice misuse, to mental disorders. Whispering is a natural form of speech for people in some social situations where privacy is desired or silence is recommended. However, for patients with aphonia, whispering is generally their primary way of communication. This can become a problem, because of the difficulty to communicate with other people, and can even cause problems in daily lives or even work related activities. There are some solutions for this problem regarding laryngectomized patients, like the use of an electrolarynx, that recreates an artificial voice, the use of esophageal speech and the tracheo-esophageal puncture with prosthesis, but all of them have some disadvantages and require some degree of practice to master speaking. In terms of technologies, there also exist silent speech interfaces, that are not yet convenient solutions. Some mobile applications also try to help with this problem, that are generally based in text-to-speech conversion. They require a text input by the user that is followed by its reproduction in speech, resulting in a slow and unnatural usage. Some of these applications operate in real-time by means of a simple click on predefined buttons with text, which also have limitations. With that in mind, the goal in this dissertation is to develop *MasterVoicing*, a mobile application, for the iOS platform, whose purpose is to give aphonics another alternative to communicate, using their natural way of communicating - whispering. Its validation is verified by the performance of usability tests and its aim is to work in real-time, integrating a whisper-to-speech conversion algorithm that reconstructs natural, voiced, speech from whispers, giving aphonics an easy tool to regain some of their communication freedom, without the drawbacks of other methods that are available.

Resumo

A afonia, também conhecida como perda de voz, é uma condição que afecta o sistema fonético humano e que se caracteriza pela inabilidade de uma pessoa produzir sons normais de fala. Esta incapacidade pode variar de grau entre a perda parcial de voz, conhecida como rouquidão, até à perda quase total de voz, sendo que neste caso a voz consiste apenas em sussuro. As suas causas podem ser físicas, relacionadas com ferimentos, procedimentos cirúrgicos ou maus hábitos, como mau uso da voz, ou causas psicológicas, relacionadas com problemas mentais ou traumas experienciados. Sussurrar é uma forma natural de comunicação para as pessoas em certas situações sociais em que a privacidade é desejada ou o silêncio é recomendado. No entanto, para os afónicos, sussurrar é geralmente o seu principal meio de comunicação. Isto pode revelar-se um problema, por causa da dificuldade de comunicar com outras pessoas, e pode até causar problemas no seu dia-a-dia ou trabalho. Existem algumas soluções para este problema relativamente a pacientes laringectomizados, como o uso de uma eletrolaringe, que recria uma voz artificial, o uso da voz esofágica e a prótese traqueoesofágica, mas todas elas têm as suas desvantagens e requerem alguma prática e aprendizagem para conseguir algo semelhante à voz normal. Em termos de tecnologias, existem também interfaces de fala silenciosa que, contudo, não são ainda soluções convenientes de utilizar. Existem também aplicações móveis que tentam ajudar com este problema, que são geralmente baseadas na conversão texto-para-fala. Elas requerem a inserção de texto por parte do utilizador, à qual se segue a sua reprodução em fala, o que resulta numa utilização lenta e artificial. Algumas destas aplicações funcionam em tempo real, através de um simples clique em botões com texto predefinido, mas têm também limitações do ponto de vista prático. Tendo isto em consideração, o objectivo desta dissertação é desenvolver uma aplicação móvel, denominada *MasterVoicing*, para a plataforma iOS, que pretende fornecer aos afónicos outra alternativa de comunicar, utilizando o seu meio natural de comunicar - sussurrar. A sua validação é verificada pela realização de testes de usabilidade e o seu objectivo é funcionar em tempo real, integrando um algoritmo de sussurro-para-fala, que reconstrói fala natural e audível a partir de sussurros, de forma a fornecer aos afónicos uma ferramenta fácil para recuperarem alguma da sua liberdade de comunicação, sem os aborrecimentos dos outros métodos que se encontram disponíveis.

Acknowledgements

I would like to express my very great appreciation to my supervisor Prof. Aníbal Ferreira for all his help, guidance and useful critiques. I would also like to thank Professor Sérgio Ivan Lopes for his initial help regarding iOS development and Professor Nuno Fonseca for providing some advice regarding specific choices about the implementation of the application.

Besides that, I would like to thank my family and Vítor for all the encouragement and patience.

Finally, I would like to give a special thanks to all the other people that I won't be naming here who contributed in some way for the concretization of this work.

Maria João Araújo Soutelo

"Every path is the right path. Everything could've been anything else.
And it would have just as much meaning."

Mr. Nobody (2009)

Contents

List of Figures

List of Tables

Abbreviations

AAC	Alternative and Augmentative Communication
ASR	Automatic Speech Recognition
CELP	Code-Excited Linear Prediction
DS	Direct Synthesis
EMA	ElectroMagnetic Articulography
EMG	ElectroMyoGraphy
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Models
iOS	iPhone Operating System
MELP	Mixed-Excited Linear Prediction
MRI	Magnetic Resonance Imaging
NAM	Non-Audible Murmur
PMA	Permanent Magnet Articulography
SDK	Software Development Kit
SSI	Silent Speech Interfaces
SVC	Statistical Voice Conversion
TEP	Tracheo-Esophageal Puncture
TTS	Text-To-Speech
UI	User Interface

Chapter 1

Introduction

For some voice disorders, as aphonia, whispering is a natural and primary way of communication. Being only able to whisper is a somewhat distressing condition that currently lacks a practical technological solution. The current mobile applications available to solve this problem are limited in terms of functionality or do not work in real time, which means for the user a slow and unnatural usage.

The work of this dissertation consists in the creation of a mobile application for iOS - *MasterVoicing* - which purpose was to include an algorithm converting whispered speech into voiced speech provided by Prof. Aníbal Ferreira, the supervisor of this work, and its posterior distribution through the *App Store*.

1.1 Context

According to a study conducted in 2005, claiming to be the largest epidemiologic study concerning the prevalence of voice disorders in the general population, "almost 30% of the adult population has experienced a voice disorder during their lifetime, and nearly 7% reported a current voice problem" [1]. Voice disorders can be a real problem to those affected, depending on their occupation. For instance, voice problems in teachers have a significant occupational impact, that result in some of them missing work and causing restrictions in work-related activities [2]. Furthermore, this condition can prevail in every kind of occupation that relates to an extensive use of the voice, such as with professional singers, actors, attorneys or salespeople [3]. They can also have significant psychological side effects, like depression, anxiety and somatic concerns [4]. One of this disorders is aphonia, also known as loss of voice. With that in mind, the aim of this dissertation is to develop a mobile application for the iOS platform to help patients with aphonia.

1.2 Motivation

Being able to only whisper can be embarrassing and even frustrating, both in personal and occupational situations. Therefore, the motivation for the work of this dissertation is, in the first place, to create a mobile application to help people with aphonia or that can only whisper, for some physical or psychological reason, with their impairment. The creation of this application is also motivated by the previous creation of another application to help to reduce the effects of a speech disorder - stuttering - also developed for the iOS platform, called *MasterPitch* [5]. Moreover, it is driven by the concern with the lack of practical solutions available to solve this problem. Lastly, there is also the desire to create an innovative application, with features that are not currently available in the applications on the Apple's *App Store* or *Google Play* store.

1.3 Objectives

The main objective of this work is to design and develop an iOS application to help people with aphonia - *MasterVoicing* - and submit it to the Apple's *App Store*, which can be divided in the following steps:

- Initial study of the problem and its current solutions;
- Further study of the state of the art about related mobile applications and methods for conversion of whispered speech to voiced speech;
- Familiarization and knowledge acquisition for the understanding of the iOS development environment and frameworks, with a special emphasis on the ones related to audio manipulation;
- Familiarization and knowledge acquisition regarding the programming languages involved in the development of the application and its algorithm (Objective-C, Swift and C++);
- Development of a throwaway prototype, that is a working proof-of-concept, which confirms the viability of the development of the application and provides a better understanding of its possible features - this also includes the performance of preliminary tests with simple real-time audio processing algorithms;
- Development of the application using a methodology adapted from an Agile software development methodology - *Scrum* - that includes the programming of all its features and the design of the user interface (UI) following the iOS Human Interface Guidelines;

- Adaptation and encapsulation in the application of the algorithm for the conversion of whispered to voiced speech;
- Verification and validation of the algorithm;
- Evaluation of the application by the performance of usability tests;
- Distribution of the application through the *App Store*.

1.4 Dissertation Structure

This dissertation is divided in five chapters. The current and first chapter (Chapter 1) contains the context, motivation, objectives and structure of this document.

The remaining chapters of this dissertation, including a brief overview of their contents, are the following:

- **Chapter 2 -** Provides an explanation of the problem and current solutions of aphonia, a comparison of existing applications with similarities with *MasterVoicing* and a brief statement and explanation of some of the methods used for the conversion of whispers into voiced speech;
- **Chapter 3 -** Contains overall information about the history, development and distribution of mobile applications for the iOS platform;
- **Chapter 4 -** Documents all the process of implementation of the *MasterVoicing* application, briefly explains the voice processing algorithm used, and presents the resulting final product and its evaluation;
- **Chapter 5 -** Includes the achievements, suggestions for future work improvements of *MasterVoicing* and conclusions of this dissertation.

Chapter 2

State of the Art

2.1 Overview

This chapter provides the definition of aphonia - the medical condition that *MasterVoicing* intends to provide an alternative solution to - and a definition, review and comparison of its current solutions (Section 2.2). Furthermore, it includes an overview about the current mobile applications available, in the two largest digital stores according to its number of applications [6] (*App Store*[1] [8] and *Google Play*[2] [10]), that are similar in terms of functionality with *MasterVoicing* (Section 2.3). Lastly, the final section of this chapter mentions some of the methods developed for the conversion of whispers into voiced speech and briefly characterizes its performance (Section 2.4).

2.2 Aphonia

Aphonia is a voice disorder[3] - a type of speech disorder[4] - that consists in the absence of a laryngeal tone, sounding like whispered speech [12]. It can have many causes, related with physical injuries or mental disorders and it can be a temporary or permanent condition.

2.2.1 Causes

Aphonia can be caused by several physical or psychological conditions [13]. This latter type of conditions are different from the former because an examination of the throat reveals normal movement of the vocal folds [14] and usually those affected cannot speak, but in some cases the patient is capable of vocalize in a whisperlike voice [15] [16]. Known examples of these

[1] Store of applications owned by Apple that sells applications for their operating system, iOS (iPhone Operating System), exclusive to their devices [7].

[2] Stored owned by Google that sells several digital content, including books, movies, music and applications that runs in devices with the operating system *Android* [9].

[3] Condition that comprises atypical or absent production of voice (usually related to alterations associated with its characteristics such as quality, pitch, volume, resonance or duration) [11].

[4] A disability related to the articulation of speech sounds [11].

psychological conditions are hysterical aphonia or selective mutism [13], where patients cannot talk because of emotional or mental causes.

Relatively to physical conditions, there are several reasons that can cause aphonia, and may involve injuries or diseases, such as ([13] [17]):

- Benign or carcinogenic tumors on the larynx or thyroid;
- Removal of some part of even the complete larynx structure due to disease[5];
- Injury after neck or chest surgery;
- Damage of the nerves that affect the larynx;
- Vocal folds paralysis[6];
- Severe laryngitis - viral, fungal or bacterial infection [19];
- Thickening of the vocal folds;
- Nodules or polyps on the vocal folds[7];
- Misuse of the voice (excessive and/or incorrect use of voice, smoking, abusive alcohol or caffeine consumption or the exposure to air pollution);
- Excessive coughing due from allergies;
- Breathing problems that affect the speech;
- Gastroesophageal reflux disease (known as acid reflux), where stomach contents flow upwards into the esophagus irritating the vocal folds;
- Primary progressive aphasia - a neurological syndrome that impairs the speech;
- Other disorders of the nervous system like spasmodic dysphonia [20], myasthenia gravis [21], multiple sclerosis [22] and parkinson's disease [23].

[5] Known as partial or total laryngectomy [18].

[6] It is usually only in one of the vocal folds, but it makes the junction of both hard and ends up affecting the voice and speech [17].

[7] Frequent among people who make a great use of the voice, like singers or teachers.

2.2.2 Current Solutions

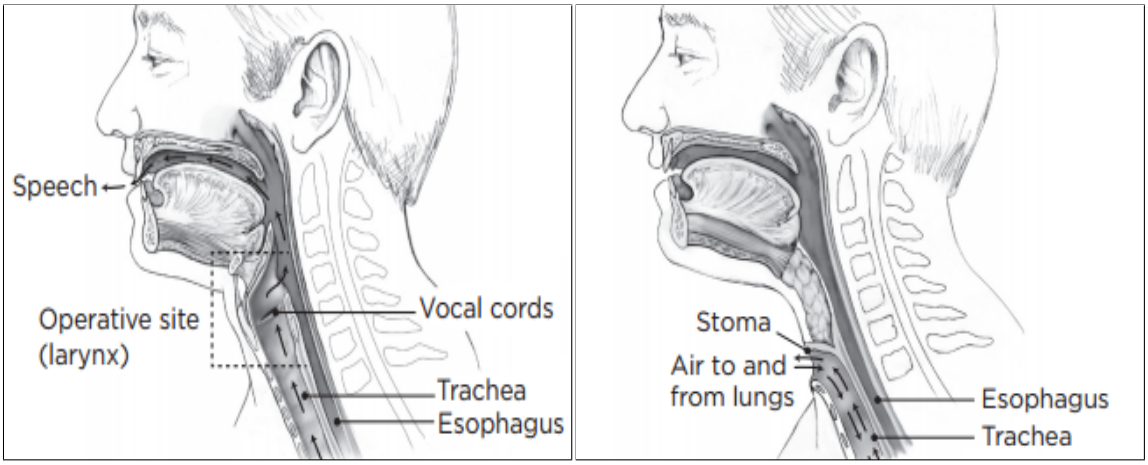
Even though aphonia has many causes, aphonics are frequently patients who have been submitted to a laryngectomy [24]. For these patients, it is usually considered by the doctors, that their best option to regain their voice is by means of speech therapy or surgical methods [25] [24]. These methods consist in learning to speak using esophageal speech or the surgical option of inserting a tracheo-esophageal puncture (TEP) with prosthesis [25]. Besides that, there are technological solutions that can be used by all patients with aphonia, such as the electrolarynges, silent speech interfaces [26] and text-to-speech (TTS) applications. There are other basic methods that aphonics can use to communicate such as writing in paper or using sign language, but the solutions presented here allow a user to reproduce speech in some way, that it is not the normal laryngeal speech[8]. Therefore, it is considered that there are currently five main solutions for aphonia, which are compared in Table 2.1:

- Esophageal speech;
- Electrolarynx;
- Tracheo-esophageal puncture with prosthesis;
- Silent speech interfaces;
- Text-to-speech applications.

For a better understanding of the first three solutions, it is important to understand the differences in terms of anatomy that a laryngectomy causes. A total laryngectomy is a complex surgery that consists in the removal of several components of the neck. According to [27] it consists in the removal of different types of body parts such as bone, cartilage, membranes like vocal folds, some rings of the trachea and even some muscles, like all of the muscles of the larynx, which possess a significant impact on the respiratory, swallowing and speaking capabilities. As this is a complicated process, we can simplify its understanding by focusing on the main components of the neck. As illustrated in Figure 2.1a, the human throat contains different components, including the esophagus, the trachea and the larynx. The esophagus is the organ through which food passes, the trachea is the tube through which the air passes to and from the lungs and the larynx contains the vocal folds, that are essential for phonation [28]. After a laryngectomy, the larynx is removed, and all the connection between lungs and mouth is cut off, that is, the esophagus and trachea are

[8] Traditional speech which involves the oscillation of the vocal folds.

no longer connected at any point (Figure 2.1b) and the laryngectomized patient breathes through a hole in the neck called stoma [28].



(a) Before (b) After

Figure 2.1: Anatomy before and after a total laryngectomy (Adapted from [29])

2.2.2.1 Esophageal speech

Esophageal speech is a form of speech where the speaker uses the esophagus as a pseudo-lung where air is stored and released in a timely fashion to produce voice [24] and it is the oldest method of voice restoration, that has been used for more than 100 years [30]. This means that it involves oscillation of the esophagus, in contrast with the tradition laryngeal speech that involves the oscillation of the vocal folds. It is a cost effective solution [30], because it only requires speech therapy sessions to learn and does not need extra devices, maintenance or procedures [25]. Even thought it gives a resemblance of the same freedom of speech as laryngeal speech, because it does not require the use of the hands, it has some disadvantages, the most important being that there can be difficulties with loudness and phrasing (the voice is often relatively low in volume and the length of sentences that can be spoken continuously are short [25]). Besides that, it is also difficult to learn for most of the patients and requires some time to master, usually in the span of months (a period of nine to twelve months),

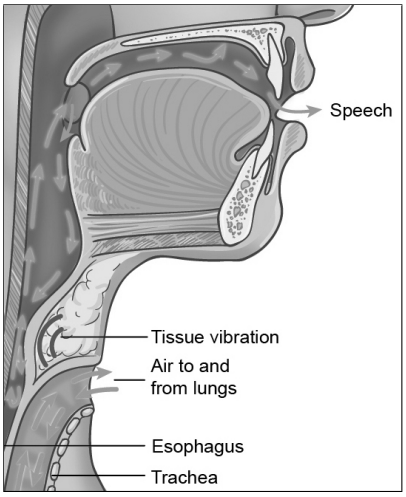
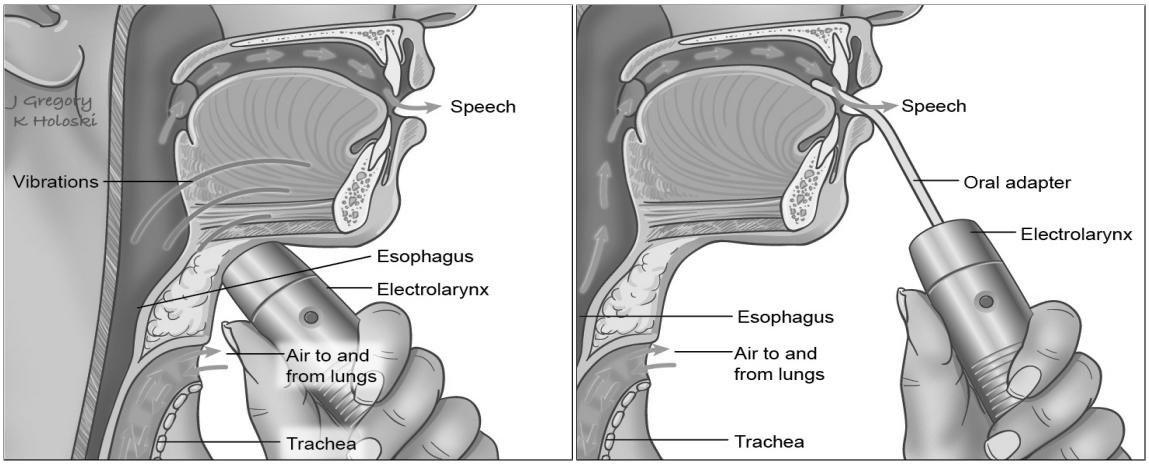


Figure 2.2: Esophageal speech (Adapted from [25])

which means that some patients prefer other solutions or just give up this method after not seeing fast results (the success rates of the development of useful speech are about 70%) [24].

2.2.2.2 Electrolarynx

An electrolarynx, alternatively known as artificial larynx, is a simple and straightforward device that allows a means of producing speech after a total laryngectomy, and whose primary advantage is that it can be used right after leaving the hospital after surgery [24]. The first electrolarynx was developed in 1859, at a time when the primary speech recovery method was esophageal speech [30]. It creates a vibration that simulates the vibration of the vocal cords and can be transformed as the patient articulates words in order to produce speech [25]. The voice produced can be of great volume, the device usage is uncomplicated and does not require much learning effort [25]. The only disadvantages are that the voice created is quite mechanical, requires the use of the hands, battery replacement or recharging and can be quite expensive [25] [31]. There are two types of electrolarynges, as shown in Figure 2.3 [25].



(a) Electrolarynx of contact (b) Electrolarynx with oral adapter

Figure 2.3: Types of electrolarynges (Adapted from [25])

The most common one is the contact electrolarynx (Figure 2.3a - also called extraoral electrolarynx [30]). This device is used by the contact with the skin of the neck, where the vibration is transmitted within the mouth through the tissues of the neck into the resonating cavities of the vocal tract [24]. The other one is the electrolarynx with oral adapter (Figure 2.3b - also called intraoral electrolarynx), that has an supplementary part that resembles a straw (also termed an oral tube) [25], where the vibration is produced directly inside the mouth.

2.2.2.3 Tracheo-esophageal puncture with prosthesis

Until 1979, when the introduction of the first approach to a surgery of tracheo-esophageal voice restoration with a prosthesis was made, the principal methods of communication after a total laryngectomy were the esophageal speech and the electrolarynx [27]. A tracheo-esophageal puncture is a surgical procedure that can be done to patients who were submitted to a laryngectomy during the same surgery (primary TEP) or in a later date in a separate procedure (secondary TEP), but the basic technique of the procedure is the same [25]. During the surgery a small hole, or puncture, that goes from the back wall of the trachea into the front wall of the esophagus, exactly behind of the trachea, is made where a prosthesis is inserted [25] [27]. This prosthesis has a unidirectional valve that lets air to flow, through the prosthesis, from the trachea to the esophagus and prevents the passage through the prosthesis into the trachea of the esophagus contents, such as food and liquids, during swallowing [25]. When the prosthesis is in place, it is possible to create tracheo-esophageal speech by covering, after breathing and while exhaling, usually with the thumb (Figure 2.4), the opening in the neck (stoma). This happens because the air passes through the prosthesis and is expelled inside of the esophagus, going up the throat and out of the mouth, creating sound, while the mucosa of the esophagus and pharynx vibrate with the air passage [25] [27].

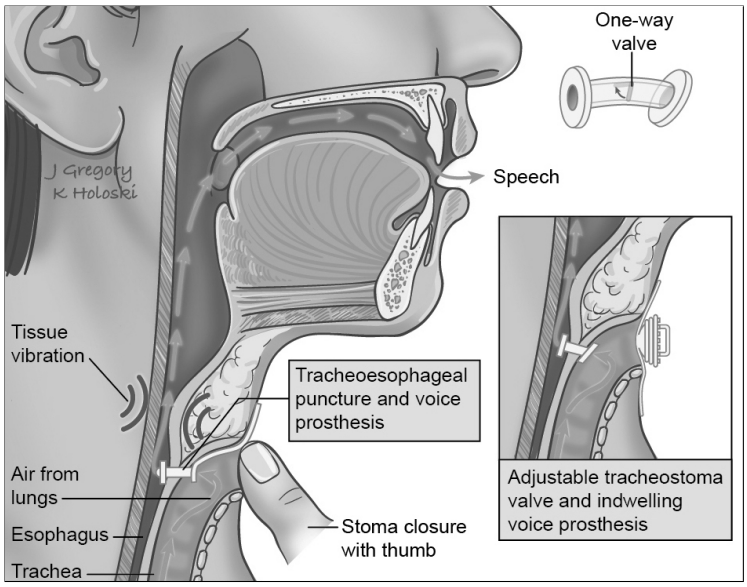


Figure 2.4: Tracheo-esophageal puncture with prosthesis (Adapted from [25])

The main disadvantage of this solution is that the prosthesis requires maintenance, such as cleaning regularly to avoid infections, so that failure in proper care often leads to health problems

[24]. Other disadvantages are that the surgery and maintenance can be expensive, since the prosthesis requires changing every few months[9] [25]. Besides that, it can also have problems related with leakage from the esophagus into the trachea and fungal infections [25]. Nevertheless, the speech created by a TEP prosthesis can be very clear and improves with practice, and there are also extra devices that allow for hands-free usage, like the adjustable tracheostoma valve (Figure 2.4) [25]. Moreover, the success rates of development of useful speech are between 80% to 90% for tracheo-esophageal speech, that is different but not necessarily superior or inferior to esophageal or electrolaryngeal speech [24].

2.2.2.4 Silent speech interfaces

Traditional speech recognition methods are based on voice elements identification that cannot be used by people with speech-related disorders, because of the lack of an audible acoustic signal [33]. By contrast, a silent speech interface (SSI) is a technology, still in development stages, that enables speech communication, by using different sensors to acquire information from the inaudible elements of the human speech production process, like the articulators[10] or the brain [35]. This produces a digital representation of speech that can be synthesized and interpreted [35] to produce artificial speech. The speech production process is thought to be the most complex motor task performed by human beings and implies the coordination of a complex set of events [34]. For a better understanding of this process, it is better to think about it considering its division into several stages, as the model proposed in Figure 2.5 suggests.

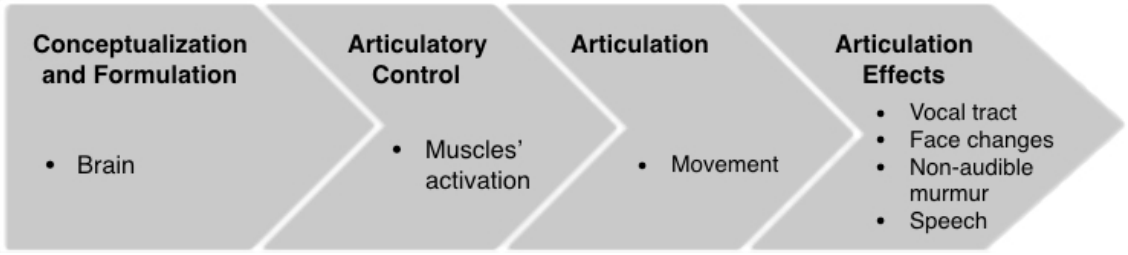


Figure 2.5: Overview of the speech production process (Adapted from [34])

It starts with the conceptualization and formulation of speech in the brain, following with the activation of the muscles related to the articulatory control, that create the movement of the

[9]There are two types of prosthesis: indwelling and non-indwelling, whose difference is that the first needs to be removed and refit by a professional and the latter can be removed and changed by the patient [32].

[10]The most important elements of the human body for speech production [34].

articulations, that result in observable and acoustic articulation effects, like the movement of the vocal tract, facial changes, non-audible murmur[11] (NAM) and speech.

Silent speech interfaces can be used for communication by people with speech disorders, or people without any problem of speech in noisy environments or when security or privacy is required [33]. Different kinds of sensors exist to capture biosignals related to speech, that are used not only to perform speech recognition without sound, but are also the core elements of several previously proposed types of SSIs [33] [36]. Regarding the decoding of the brain activity related to the speech process, significant work has been done, but with limited success. In terms of articulatory control, there are some SSIs that use the sensing of the electrical activity of the muscles related to the speech process, such as the surface electromyography (EMG)[12] [36]. Others use the sensing of the movements of the speech articulators, such as the use of non-audible murmur NAM microphones [33], electromagnetic articulography (EMA)[13], permanent magnet articulography (PMA)[14], magnetic resonance imaging (MRI), radar, video and ultrasound or a mix of both [36].

All of this speech-related information captured by these approaches is then utilized to find out the digital acoustic signal related with the desired speech information [36]. This is usually done by resorting to automatic speech recognition (ASR) to decode the information collected by the sensors and then using a TTS synthesizer for the generation of the final recreated artificial voice from the recognized text. However, this method has some disadvantages, that brought up the creation of an alternative approach, called direct synthesis (DS). As the name suggests, this approach consists in the direct speech synthesis from the information gathered by the sensors to the corresponding acoustic signal without an intermediate recognition step [36]. Comparing with the "recognize first and synthesize later" approach, DS has some advantages like not being limited to a specific vocabulary, being language-independent and allowing real-time speech synthesis [36]. Particularly for the laryngectomized, this can involve simultaneously recording both data using different SSIs' sensors and the patient's voice before laryngectomy[15], since the sensors' data alone do not offer sufficient information about the speech characteristics. Then, after the laryngectomy, all this information is used to generate artificial speech [36]. If it is only possible to record the

[11] NAM sounds are sounds with low amplitude created in the vocal tract, that result of the resonance that air produces while it passes through the larynx [33].

[12] It is called "surface" because it uses non-invasive electrodes, unlike conventional EMG [35].

[13] Monitorization of the motion of a set of stationary points inside of the vocal tract [35].

[14] Capturing the magnetic field generated when a user "speaks" by using several magnets attached to the articulators [36].

[15] The recording of the voice prior to surgery is also done for the use in TTS applications, using services like "*my-own-voice*", to create a synthetic voice that captures characteristics of the original voice [37].)

voice before surgery, the patient is asked, after the surgery, to mime speaking along to their own voice to gather the necessary sensorial information [36]. In other cases, where no information is gathered before the surgery, patients can mime along to a "donor voice"[16] that ends up being the artificial voice they speak with [36].

According to some sources [35], some of these solutions work accurately in silence and in noisy environments, and can be used by patients with aphonia and are low cost in terms of production. However, there are still problems regarding the fact that they are not quite ready for the market and are still very obtrusive for the user in terms of the devices that they need to wear for them to work (most of these systems require the use of contact sensors that are unpleasant and uncomfortable or use optical systems susceptible to external and ambient interferences [33]). However, that may change in the future, with some new research and development about contactless silent speech recognition systems[17] [33] [35].

2.2.2.5 Text-to-speech applications

Text-to-speech applications are applications that use a device, usually a smartphone, a tablet or a computer, but can also have their own specific hardware, known as TTS devices, that allow a user to read text out loud. These applications reproduce artificial speech from text, usually text inputted by the user, but can have more features. They are the most common solution for common people to work around the problem of aphonia.

In these applications, the most common and basic feature operates as follows: the user inputs what he/she wants to communicate in the form of text and the application speaks for him/her at the push of a button (features in color red in Figure 2.6 - the two features on the top right of the figure).

For smartphones and tablets, there are other extra features (as an example, the features in color blue in Figure 2.6 - the two features on the bottom right of the figure and left) that can be found in this type of applications like:

- Favorite phrases: the ability to add phrases to a list of favorites, with the main purpose of using them frequently[18] [39] [40];

- Phrases history: a record of the previously "spoken" phrases [38] [39];

[16] Patients can request a family member or friend to have their voice synthesized and then use it for themselves [37].
[17] Like the proposed application to implement a contactless SSI using as a sensor an impulse radio ultra-wide band (IR-UWB) radar [33].
[18] Some can be organized in categories based on location, audience, and situation [38].

- Settings for the characteristics of the artificial voice, like the volume, pitch, rate, gender, language and speaker[19] [39] [40] [42] [43] [44];
- Save text as audio file for later reproduction (for example in a presentation) [43] [45] [46];
- Buttons with text, sorted by category (customizable [47] [48] or predefined [49]);
- Prediction of the most likely next word the user is going to write [50] [51];
- Speaking each word, sentence or new line as it is typed [40] [50];
- Clear text after speaking [39] [40];
- Reading out loud the text of files in several formats like "*.pdf*", "*.epub*" or "*.docx*" [42] [52];
- Reading out loud Internet's webpages by fetching its text content given their URL [53].

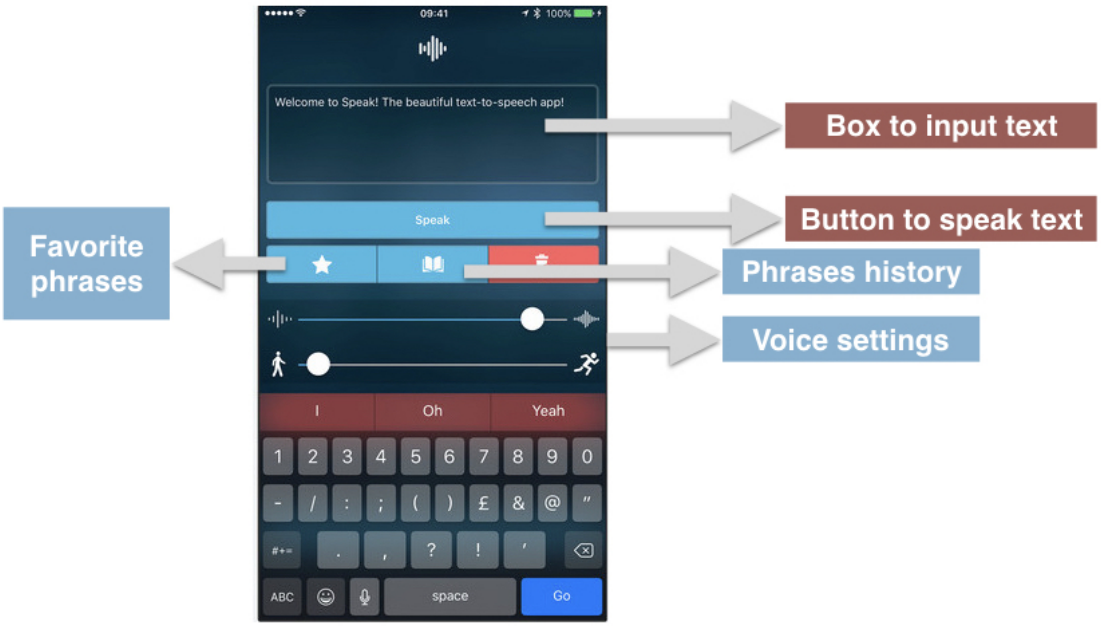


Figure 2.6: Screenshot of a TTS iOS mobile application - "Speak - Text To Speech" (Adapted from [39])

Besides that, there are TTS devices, that usually have even more features, such as the ability to send emails and text messages [54] [55]. Some of them also have features typically found on computers and smartphones such as the access to social networks, the ability of taking and sharing photos, browsing the Internet or playing games. In most cases, they are considered to be

[19] Some applications have different speakers for the same language [39] [40] [41].

alternative and augmentative communication (AAC[20]) devices, and can be a good communication tool for aphonics, thanks to their features. The main advantage between using these devices and a general TTS application in a smartphone is that the hardware of these devices is usually designed with the intent of facilitate communication. This means that the hardware of TTS devices can have features that, like smartphones, unusually have. For example, good speakers (Figure 2.7a) can help making the artificial speech audible in places with noise [57] and a dedicated real keyboard (Figure 2.7b) can help those who prefer to type in real keys instead of a touchscreen [58].

(a) "Tobii Dynavox T15" (Adapted from [57]) (b) "DynaWrite 2.0" (Adapted from [58])

Figure 2.7: TTS devices

In terms of usability, there are what seem to be very good applications, according to their reviews, that use the predefined buttons category mixed with button personalization [47] and possibility of eye control [59], which are probably the fastest way available, in terms of technological solutions, for a person that cannot speak to communicate. Another advantage of these applications is that most of them are free and are available in different platforms and devices. However, they can also be expensive [47] [54] [57] and do not match the same freedom that *MasterVoicing* intends to provide its users, by allowing them to speak in real-time without adding any input besides their own "voice". Another disadvantage of these applications is that they are only as fast as its user is at inputting text. For that reason, they are not a very convenient solution for real-time conversations. Besides that, it is also important to note that, according to UNESCO[21], almost 17% of the world's adult population is not able to read nor write [60]. Therefore, this is not a solution for everyone. Moreover, the main important features found in these applications can be scattered around, that is, to get all the functionalities cited above, the user may have to download several applications and use different devices[22], which is also an inconvenience.

[20] Consists in all means of communication, besides oral speech, used to express thoughts [56].
[21] United Nations Educational, Scientific and Cultural Organization [60].
[22] Because there exist for example applications for tablets only [50].

Table 2.1: Comparison between current solutions for aphonia [24] [25]

Solution	Advantages	Disadvantages
Esophageal speech	– Does not require devices or procedures – It is cost effective – It is hands-free	– Difficult to learn for most patients – Requires some time to master optimal voice quality – The voice is often low in volume and the length of sentences that can be spoken continuously are short
Electrolarynx	– Can be used right after surgery thanks to its simplicity of usage – The voice produced can be of great volume	– Produces a mechanical artificial voice – Requires maintenance and it is not hands-free
TEP with prosthesis	– Speech can be very clear and improves with practice	– The prosthesis requires regular cleaning and changing, can fall out and it is not always hands-free – It is usually expensive
Silent speech interfaces	– Do not require actual speech, only speech movements – Work in places with a lot of noise	– Not yet ready for commercial uses – Need more research and development to work consistently and accurately
Text-to-speech apps	– Many are available for free – A better solution than other simple writing communication methods	– Can be expensive if the user wants a dedicated device or more functionalities – Not very convenient solution for someone to have a conversation in real-time (are only as fast as its user is at inputting text) – Require that its user knows how to write

2.3 Related Mobile Applications

In terms of mobile applications, a thorough search of the Apple *App Store* and the *Google Play* Store[23] show that there is not any mobile application to assist people with aphonia, with their sporadic or permanent disability, that uses whisper-to-speech conversion. However, there are applications that can be related to the application developed for two reasons: the first is that they try to solve the same problem and the other is their similarities in terms of functionality, even thought being used for other purposes. Related to the first reason are the TTS applications, mentioned in the previous section (Section 2.2.2.5). The other type of applications found on the applications' stores that are related to *MasterVoicing* for the former reason can be sorted in the following categories:

- Voice translation;
- Voice changing;
- Sound amplifying.

All of these applications record and process speech. Some of them use it to retrieve information and others only to transform its audio signals in some way. The similarities and differences of these three types of applications with respect to the *MasterVoicing* application are summarized in Table 2.2 for a quick comparison, after the three following sections that contain more detailed information about each type.

2.3.1 Voice Translation Applications

Voice translation applications [61] [62] [63] are applications that, as the name suggest, translate spoken words between different languages. They work by recording what a user says in a given language and give the corresponding translation of that same recording in another language. They do not actually work in real-time, that is, they are not always recording what the user says. This is one of the main differences between these applications and the *MasterVoicing* application. The process of functioning of these applications is recording the voice of the user, processing the sound to identify the spoken words, translate them and then use a TTS synthesizer with the correct audible translation. Another difference is that this process is language dependent, and needs to process the actual words the user say, while *MasterVoicing* is language independent. Finally, the

[23] This search was narrowed to these two stores because they are the ones which have the higher number of applications available for download [6].

last main difference is that they do not actually change the recorded speech, they only use it to process its meaning, so that the final result is not a changed version of the original recorded voice, being instead an artificial voice produced by a TTS synthesizer.

2.3.2 Voice Changing Applications

Voice changing applications work in a similar way as voice translation applications, in the way that they record what the user say, but to a different end - to allow it to change the recorded audio with different sound effects. However, they are different from the previous applications in the sense that they do not require the processing and understanding of the speech that is embedded in the recordings. Also, the sole purpose of these applications is recreative, not having an actual useful end as the previous applications, hence the sound effects are, for example:

- Based in changes of the basic characteristics of the voice, like pitch or speed [64];
- Robotic, "ghostly", helium voice, alien and monster alike voice effects [65] [66] [67];
- Changing the voice to sound like the voice of a celebrity or fictional character [65] [66].

They are marketed as applications for voice changing that allow to save the recorded voices just for fun or for example to make prank calls[24] [68] [69] [70] [71].

Most of these applications also do not work in real-time [64] [72] [73] or, if they do, usually require headphones [66] [74]. Nevertheless, there are cases of these applications that work in real-time and without headphones, but may not do it very successfully, according to certain reviews [75].

2.3.3 Sound Amplifying Applications

Lastly, sound amplifying applications have another main purpose - to help people with hearing problems [76] [77] [78] [79] - even thought they are sometimes marketed by its programmers for another purposes, like "auditory espionage" [80]. They typically record the surrounding sounds using the microphone of the mobile device and then process and enhance them to provide an easier and clearer audition to the user [81]. Just like *MasterVoicing*, they work in real-time, and are always recording and producing sound, but require headphones to work (the reason for this may be avoiding problems related with feedback or simply because the usability of these applications makes more sense with them).

[24] Even thought this requires extra charges to the user [68] [69] [70] [71].

Table 2.2: Similarities and differences of the related mobile applications

Application Type	Similarities	Differences
Voice translation	– Work without headphones	– Do not work in real-time – The spoken voice has no resemblance to the original recorded voice
Voice changing	– Some work in real-time – Some work without headphones	– Have only recreative purposes
Sound amplifying	– Work in real-time – Some exist with the intention to help people with disabilities	– Need headphones to work

2.4 Signal Processing Methods for Converting Whispered Speech into Voiced Speech

Currently, the majority of technological systems related with speech communication, such as ASR systems, operate exclusively with audible and vocalized speech. This means they are incapable of operating with whispered speech or do it rather poorly [82] [83]. The reason for this to happen is that, according to its nature and mechanism of production, whispering is considerably different from voiced speech [84]. Whispered speech is characterized by "a noisy structure" and the lack of sound production that involves moving the vocal folds [84] [85]. For this reason, whispering does not have the same fundamental frequency of voice, pitch contours and other important features [84] [85]. Also, the power of its sound spectrum is weaker compared to voiced speech, which makes it more susceptible to the interference of ambient background noise [82] [84] [85]. Due to these differences, processing and reconstructing whispered speech is more challenging than voiced speech [85], which include two possible approaches [82] [83]:

- Work directly with the whispered voice input;
- Work with a "speech-like" form of signal created from the whispered voice input.

The latter approach may be preferred and it is usually termed "speech reconstruction" or "conversion of whispers into natural speech" [82] [83].

The conversion of whispered speech to voiced speech can be done resorting to several methods [82] [86] [87], such as:

- Mixed-Excited Linear Prediction (MELP);
- Code-Excited Linear Prediction (CELP);
- Statistical Voice Conversion (SVC);
- Other new approaches or hybrid or improved versions of MELP and CELP.

The first two methods consist in voice coders (also known as vocoders[25]) that were originally designed for the transmission of compressed speech, but are now used for the purpose of relating whispered and voiced speech signals [87].

2.4.1 MELP and CELP Approaches

MELP is the oldest of the vocoders and reportedly works considerably well [82]. However, it requires training using the original voice of the speaker before performing whispers to voiced speech conversion. This means that it cannot be used when it is not possible to record the original voice of the speaker [82] [86]. Also, it is not appropriate for real-time operation [82].

CELP introduces the use of an "excitation codebook" [87], with the purpose to remove the requirement of a priori speaker's original voice information [86], and provides a more natural synthetic speech [82].

Both MELP and CELP work well for phonemes[26] and single words, but results are poor and lack evaluation for continuous speech [82] [83]. Moreover, they have a considerable computational complexity [83]. These two approaches work by decomposing whispers into components and then reconstructing speech with transformed (or improved) pitch information. The idea behind this is that whispered speech is similar to speech with no pitch and therefore it can also be decomposed just like fully voiced speech [82].

2.4.2 Statistical Voice Conversion

Another approach, that was created more recently, is the Statistical Voice Conversion (SVC). As the name suggest, this method uses statistical methods - making use, for example, of Gaussian Mixture Models (GMMs) [90] - to model pitch and parameters of the speech from parallel whispered and voiced speech training information [82]. Some of the implementations of these methods

[25] "An electronic device that synthesizes speech" [88].
[26] Small set of speech units by which words and sentences are formed [89].

use NAM microphones, that capture body-conducted unvoiced speech and have some advantages like being immune to external noise disturbances and providing an effective conversion [86]. Even so, they have some disadvantages, such as not taking into consideration some information that is suppressed in the process of capturing information, not being suited for real-time operation [86] and involving a high computational complexity [83].

2.4.3 Other Approaches

Other approaches, new, or based on the previous methods [91], have also been introduced, with the main difference that some of them do not require a priori speaker information (do not require parallel original speech and whisper input information) [87]. Also, they have other advantages such as a lower computational complexity, compared to the previous ones, are designed for real-time operation and improve the reconstruction quality of the whispered voice [82] [87].

2.5 Conclusions

In order to create an application that can help aphonics, it is necessary to understand the problem of aphonia, as well as its current solutions, their problems and their limitations. This chapter gives some insight into this matter as well as the current state of what the current mobile applications' technologies can offer in terms of features. According to the research fulfilled, two conclusions are that even thought there are currently some different solutions for aphonia, that have been improved and created over the years, they all have different disadvantages and some need further development. It is proposed that a new solution lies in the development of a mobile application that converts whispers to speech, combining, in a certain way, the technologies of mobile applications and the methods for reconstruction of natural speech from whispered speech, also referred in the last section of the chapter.

Chapter 3

iOS Overview

3.1 Introduction

iOS is the operating system that runs on *iPhone*, *iPad* and *iPod Touch* devices, managing its hardware and providing the required technologies for the implementation of native apps [92].

This chapter gives a brief overview about iOS history, development and application distribution. Apple provides a more thorough and detailed insight about the content addressed in Sections 3.3 and 3.4 in the *Apple Developer* website [93]. This chapter will focus only on information related with iOS, even though it shares some aspects with other operating systems used by different Apple devices, such as *macOS*, *tvOS* and *watchOS*.

3.2 iOS History

Before actually getting into the theme of iOS development, it is important to understand a bit of this operating system's history and evolution.

The creation of iOS was marked with the introduction of *iPhone*, in January of 2007 [94]. The current Apple devices' operating system didn't have a specific name until March of the same year, when the beta version of the first Software Development Kit[1] (SDK) was launched for the *iPhone* [96]. After that, it was known as iPhone OS [96].

In 2008, Apple introduced the *App Store*, a place where anyone could sell an app for a price of their choosing [94]. This happened the year after Apple first told developers that the development of applications in *iPhone* would be made by creating web applications for *Safari* [97], its native web browser. Since that, iPhone OS changed and evolved over the years, and its name was changed in June of 2010, with the introduction of iOS 4 and the *iPhone 4* [96]. Over the years, the *App Store*

[1] A collection of frameworks (libraries, headers, and resources) that represent the API for a specific operating system version [95].

has been proving to be the right decision, by currently having 2.2 million applications available [98] and generating billions in profits for both Apple and developers [94].

In the following years, with each new version of iOS, many features were introduced, with much or less success, such as *iMessage* (texting over data), *Siri* (Apple's voice-activated virtual assistant [94]), *Apple Maps* and *Passbook* (a virtual wallet, later renamed to *Wallet*). With iOS 7, and the release of the *iPhone 5S* and *iPhone 5C*, Apple unveiled a totally new UI design, that promoted a new flatter aesthetic, in detriment of an skeuomorphic one [94].

Since iOS 9, Apple started to devote its developing resources towards improving the overall stability of the operating system and adding smaller, but useful, features, to improve overall usability [94].

With the newly announced iOS 11, scheduled to be released in the Fall 2017, Apple will possibly make some of the biggest enhancements to iOS, both related to significant new features (including all new augmented reality and machine learning frameworks) and overall minor refinements [99].

3.3 iOS Development

The most important prerequisite needed to develop a native iOS application is *Xcode*, presented in Section 3.3.2, that, together with a basic understanding about the related technologies (Section 3.3.1) and the tools it contains, can help make better choices about how to design and implement iOS applications [92].

3.3.1 Technologies

Apple delivers most of its system's interfaces in special packages called frameworks. A framework "is a directory that contains a dynamic shared library and the resources (such as header files, images, and helper apps) needed to support that library" [92]. To use frameworks, they have to be added to the application's project on *Xcode*. The architecture of iOS contains those frameworks, which can be viewed as a set of layers, as shown in Figure 3.1 [92]:

- **Cocoa Touch -** contains fundamental frameworks, most importantly the ones that define the appearance of the application [100];
- **Media -** contains the graphics, audio and video technologies used in the application [101];
- **Core Services -** contains fundamental system services of the application [102];

- **Core OS -** contains the low-level features that most higher-level technologies are built upon [103].

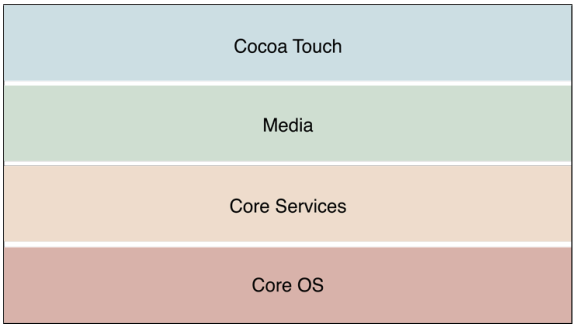


Figure 3.1: iOS architecture layers (Adapted from [92])

The lower layers contain fundamental services and technologies; higher-level layers provide more refined technologies and services [92].

The initial versions of iOS were designed to support binary files on devices using a 32-bit architecture. In iOS 7, support was introduced for 64-bit architecture and the release of iOS 11 will set that the 32-bit architecture will no longer be supported. This was pushed by Apple because applications may run faster when compiled for the 64-bit runtime, because of the availability of extra processor resources in 64-bit mode [103].

3.3.2 *Xcode*

Xcode is a complete developer toolset used to create applications for iOS [95], and includes the SDK that enables the creation of applications that run on specific versions of iOS [104].

Xcode packages the iOS applications as bundles[2] [105]. A typical application bundle has different types of files: an *Info.plist*[3] file, an executable file, resource and other support files [106].

The basic utilities of *Xcode* are the *Interface Builder*, the *Debugger* and the *Documentation*. The *Interface Builder* (Figure 3.2) offers a graphical environment for building the UI of the application [107], the Debugger (Figure 3.3) aids the process of finding and eliminating problems in the applications' code [108] and the *Documentation* provides most of the information needed for the development [109].

As part of the *Xcode* toolset, there are another two main tools: *Simulator* and *Instruments* [110] [111].

[2] A bundle is a directory in the file system that groups related resources together in one place [105].
[3] "A structured file that contains configuration information for the application" [106].

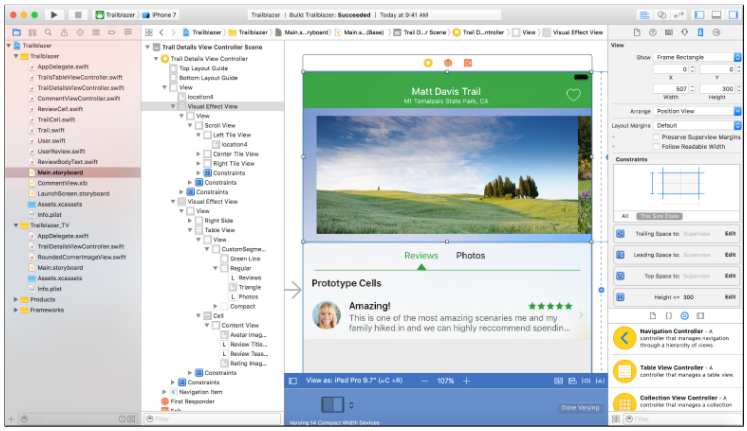


Figure 3.2: *Xcode Interface Builder* (Adapted from [112])

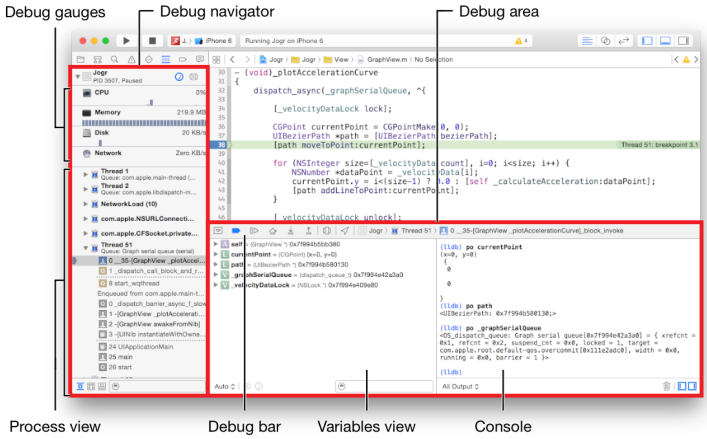


Figure 3.3: *Xcode Debugger* default layout (Adapted from [113])

The *Simulator* tool (Figure 3.4) provides a simulation of iOS, from the most recent version to the legacy versions, for testing a build of an application without needing an actual physical device, but not as reliable [110].

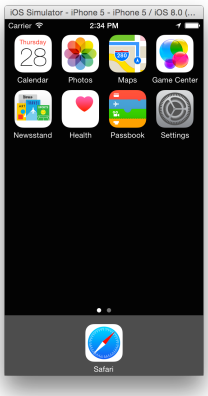


Figure 3.4: *Simulator* tool (Adapted from [110])

The *Instruments* tool is designed to help developers, by providing testing and performance-analysis of an application [111].

For testing purposes, there are also a test navigator designed to facilitate the process of creation, management, running and reviewing of code tests [114].

3.3.3 Programming Languages

There are two choices of native programming languages for iOS development: Objective-C and Swift [115]. Objective-C is an older language, designed in the early 1980s, while the creation of Swift only began in 2010. Objective-C became the foundation of Apple products prior to 1996, while the first version of Swift was released in 2014 [115].

Swift and Objective-C projects are nearly identical. The main difference between them is that Swift has no header files. In terms of interoperability, it is possible to use Objective-C APIs in Swift and vice-versa, which is also an advantage. However, unlike Objective-C or C, it is not possible to import C++ code directly into Swift; it is necessary to create an Objective-C or C wrapper for the C++ code [116].

The fact that Swift is a "still evolving" language, contrary to Objective-C, means that every language update requires updating the application's code to the new version [115]. However, the frequent updates add to the belief that Apple and the developer community are committed to Swift [115]. Another difference between Swift and Objective-C is that the former only supports operating systems prior to iOS 7 [115], but that is likely not a worrisome problem, since only 3% of devices currently use a lower version than iOS 9 [117].

Swift is stated as a powerful and modern programming language that is also easy to learn, and it is designed for safety, performance and software design patterns [118] [119]. Its purpose is to combine the best aspects of modern languages with the wisdom from its open-source community and Apple expertise [120]. At the end of 2015, the Swift language and all its supporting components were published as Open Source[4] [119].

Apple has made many efforts to promote the learning of the new language by creating helpful guides, documentation and learning tools like the *playgrounds*[5], therefore, Swift seems to be the best bet for the future of iOS development.

[4]For reference, there is a complete explanation of the definition of Open Source in [121].
[5]Small programs that can instantly show the results of its written code [122].

3.4 iOS Distribution

The distribution of applications in the *App Store* requires a membership on the *Apple Developer Program*. Besides the main benefit to distribute applications through the *App Store*, the *Apple Developer Program* gives access to exclusive software and tools, such as operating systems' beta releases, advanced app capabilities, testing and support [123]. Enrolling in the *Apple Developer Program* requires an *Apple ID* and an annual fee. Developers can enroll as an individual or as an organization [124].

The *App Store* makes the process of distributing an application easier, since it provides a focused channel for users to discover, purchase and download applications. Moreover, it handles worldwide payment processing and does not charge hosting fees, giving developers at least 70% of sales revenue [123]. With the membership, it is even possible to beta test the applications with a limited number of users before rolling it out to the complete *App Store*'s users [123]. For the process of submission and management of the applications for sale, the *Apple Developer Program* provides access to a suite of web-based tools, called *iTunes Connect* [125].

The applications submitted in *iTunes Connect* are all reviewed, to make sure that they comply with a set of rules and guidelines, described in *App Review* [126], and require approval before being put on sale on the *App Store* [125].

3.5 Conclusions

iOS and its development process have evolved over the years, with new operating system features and developer tools improvements. Before iOS, there was no defined mindset of "Everyone Can Code" [127], but today this philosophy seems to be increasingly more popular and accepted, which means that developing tools and programming languages are becoming simpler, but without compromising power. Moreover, the easy access to application's distribution its seemingly proving to be a good means of creating new jobs for single developers and small mobile development companies.

Chapter 4

MasterVoicing Application

4.1 Introduction and Overview

This chapter documents all of the steps and information related with the creation of *MasterVoicing*. It contains four main sections, that follow the current one, which provide a better understanding of the process: Implementation (Section 4.2), Algorithm (Section 4.3), Final Product (Section 4.4) and Evaluation (Section 4.5).

The first section includes all the information related to the learning process, initial tests, throw-away prototype and actual development made in the implementation of *MasterVoicing*. It takes into account all of the chosen paths for the development and design of the application, including its reasons, and also the experienced problems and how they were tackled and solved (or unresolved).

The next two sections include an explanation and presentation of all the features of the final product of this dissertation, the iOS application developed - *MasterVoicing* - and a brief explanation of the encapsulated algorithm.

The last section includes all the tests performed for the evaluation of the application.

4.2 Implementation

The current chapter, related to the implementation of *MasterVoicing*, can be divided in three main parts, not explicitly shown in the organization of the following sections. The first part includes Section 4.2.1 and Section 4.2.2, which provide a better understanding of the software development process. The second part includes four sections, where all of the initial learning process that resulted in a working prototype is documented: Section 4.2.3, Section 4.2.4, Section 4.2.5 and Section 4.2.6. Finally, the last part includes the last section, Section 4.2.7, that covers all the steps of development of the final product (portrayed in Section 4.4).

4.2.1 Software Development Process

Software development, especially at the professional level, can be complex and have many challenges [128], which is of the reasons why software development processes were created and developed.

A software development process is an outline for the activities necessary for software production [129] [130]. The adoption of a software development process can have many advantages, related with improvements in productivity, and the reduction of costs related with production, future maintenance and improvements of the resulting product [129] [130].

There are four fundamental activities related with the process of developing software [130]:

- Specification: definition of the wanted capabilities and functionalities of the software;
- Design and Implementation: developing of the software based on the specification;
- Validation: testing to confirm that everything works as the specification;
- Evolution: addition of new features and maintenance of current ones.

These steps were surely necessary for the creation of *MasterVoicing*, and are documented in the following sections of this chapter, whose implicit relationships are suggested in Table 4.1.

Table 4.1: Relationship between sections and software development process activities

Activities	Sections
Specification	– 4.2.2 Software Development Methodologies – 4.2.3 Initial Study – 4.2.4 Initial Requirements Analysis
Design and Implementation	– 4.2.5 Initial Tests – 4.2.6 Throwaway Prototype – 4.2.7 Application Development
Validation	– 4.5 Evaluation
Evolution	– 5.2 Future Work

Every software development process includes these four activities in some way, even if they use different approaches to each one, and is usually labeled as one of two types: "plan-driven" (traditional approach) or "agile" (modern approach) [130].

4.2.2 Software Development Methodologies and Framework

A software development methodology is a particular approach to a given software development process, that comprises a specific and well defined set of tasks and techniques for the management and completion of a software product. There are several methodologies with different characteristics, according to how they handle the different stages of the software development process.

Agile based methodologies and frameworks were developed due to dissatisfaction with the heavyweight approach of the plan-driven ones [130]. This traditional methods resulted in more time being spent in the specification activity (Table 4.1) than all of the following ones [130]. This was considered by many a sluggish approach, and a waste of precious time for development, hence the name given to the modern approaches, which intended to fasten the whole software development process, was "agile" [130].

According to a recent global survey [131], Agile development is currently very popular, and several companies, from different industries, are actively using it or trying to adopt it. For this reason, its type of methodologies seemed to be the most recommended one, and was the chosen for the development of the application. However, as a way to better understand the possible capabilities desired in the application, it was thought it was a better approach to beginning with a software prototype, before the actual final development.

Sections 4.2.2.1 and 4.2.2.2 contain a brief explanation of the software development methodology and framework related with the implementation of *MasterVoicing*, the former for being used as a first approach and the latter for serving as a basis for the final product's development.

4.2.2.1 Software Prototyping

Software prototyping consists in the creation of an incomplete demonstration of the desired final product.

Prototypes are often used in development, and can be useful, both *low-fidelity* and *high-fidelity*[1] ones. They are usually used for two main reasons and in different phases of the development process [130] [133]:

- In the specification, to better understand the requirements before the actual design and programming of the application and validate its implementation;
- In the design, to explore and define the UI design.

[1] *High-fidelity* prototypes provide more interactivity, have more realistic visuals and include all the content that would appear in the final product, being as close as possible to its final design [132].

There are two main types of software prototypes [133] [134]:

- Throwaway - used to validate or define the requirements and discarded after that for the development of the final product;
- Evolutionary - based on building an actual functional product, with minimal functionality, so that can be later improved upon to result in the final product.

Prototypes can also be developed using two different approaches: vertical or horizontal. A horizontal prototype is used mainly for the understanding of the UI design, since it gives a broader view of the entire final product, without going in depth or detail about internal functionalities [133]. On the contrary, a vertical prototype is used to demonstrate a detailed elaboration of a specific functionality or set of functionalities [133]. Therefore, horizontal prototypes are better for getting an overview of the final product as a whole, whereas vertical prototypes are useful for getting in exact detail of the technical aspects of a given functionality [133].

The initial phase of the implementation of *MasterVoicing* was the development of a prototype of the application, that include its core features. The characteristics of this prototype are better explained in Section 4.2.6.

4.2.2.2 *Scrum*

Scrum is a framework for dealing and solving complex tasks, created with the purpose of delivering software products productively, in an adaptable manner [135]. It consists of a set of values, roles, events, artifacts and rules, better explained in Table 4.2 [135]. Each one of these components serves a particular purpose and they are necessary for the success of its usage and results [135]. While sometimes called a methodology, it does not have the detailed and formal structure of one, and has the purpose to ensure that all of the process is accounted for transparency, inspection and adaptation, with a set of values everyone needs to follow [135]. Based on a recent global survey [131], it can be said, with a high degree of confidence, that *Scrum* is currently the most widely used Agile implementation.

This framework is supposed to be lightweight, simple to understand, but is not necessarily easy to master, which is one of the reasons the role of *Scrum Master* exists [135]. The *Scrum Master* is the one responsible for making sure that the framework process is understood and put in practice properly by the whole team, with the application of all its components [135]. The *Development Team*, as the name implies, are the individuals responsible for creating the actual accountable product, which creation is managed by the vision of the *Product Owner* [135].

Table 4.2: *Scrum* components [135]

Type of Components	Components
Values	– Commitment – Courage – Focus – Openness – Respect
Roles	– Product Owner – Scrum Master – Development Team
Events	– Sprint – Sprint Planning – Daily Scrum – Sprint Review – Sprint Retrospective
Artifacts	– Product Backlog – Sprint Backlog

Finally, we have the events and artifacts. They are used with a set of rules and tactics that are usually adapted according to different constraints, such as the type of project or size of the team. The first steps in the process are filling the *Product Backlog* and planning a *Sprint*, in the initial *Sprint Planning* meeting. The *Product Backlog* holds everything that the product requires to be considered "done" and the *Sprint Backlogs* have the specific tasks chosen to be completed for the closing of the correspondent *Sprint*. A *Sprint* is a planned set of days, preferably no more than a month, in which the *Development Team* works in a potentially usable product, meant to be incremented with each consecutive *Sprint*. During a *Sprint*, there are daily fast meetings for the *Development Team*, called *Daily Scrum*, to coordinate activities and create a plan for the next day. After the conclusion of a *Sprint*, there are two meetings with different purposes: *Sprint Review* and *Sprint Retrospective*. In the *Sprint Review*, all the stakeholders of the project inspect the completed work on the *Sprint* and adapt the *Product Backlog* if necessary. In the *Sprint Retrospective* they focus the inspection on other aspects, like the team relationships, the process and tools used, and identify possible improvements and adaptations to make for the next *Sprint*. [135] [136]

4.2.2.3 Adopted Methodology

Since the creation of software development methodologies stem from the goal of better managing teams of professionals, that develop software on a professional level, they usually are not inherently design to be used by a single person. This is, for example, noticed in *Scrum*, with the existence of different roles in a team and daily meetings, explained in the previous section. For this reason, it was necessary to adapt a methodology for personal development, to be used for the development of the *MasterVoicing* application.

The software methodology used was created using an adapted version of *Scrum*, also loosely inspired by two other proposed methods of a software development method for single developers, which are based in Agile development methodologies: *Cowboy* [136] and *Scrum Solo* [137].

The general idea was to take some of the advised steps of these methodologies, choosing the ones that make the most sense for the development of the application, and design a personal adapted methodology, with all the advantages of the Agile philosophy, but only one team member. The resulting components of the methodology are shown on Table 4.3.

Table 4.3: Adapted methodology components

Type of Components	Components
Roles	– Product Owner – Single Developer
Events	– Initial Requirements Meeting – Sprint – Sprint Planning – Sprint Review – Sprint Retrospective
Artifacts	– Product Backlog – Sprint Backlog – Code

The main differences from this adapted methodology with the official *Scrum* methodology is that there was no *Scrum Master* or *Daily Scrum* meetings, since it was intended for personal development. However, there was a *Product Owner* – This works' supervisor. The duration of the

Sprints were also thought to be less than the span of two weeks, to adapt to the development time constraints and distribute the work more evenly.

The *Product Backlog* was populated with *User Stories*[2] and *Epics*[3], based on the information gathered in the *Initial Requirements Meeting* (Section 4.2.4) and during the creation of the *Throwaway Prototype* (Section 4.2.6), that took place before the final application development (Section 4.2.7).

The *Sprint Planning* was made one day before the beginning of each *Sprint*, and focused on choosing *Epics* for each one by tackling the most important *User Stories* first, so that it would be possible to have a working product as soon as possible, and choosing the following *Sprint* duration. In order to clarify the importance of each *User Story*, they were attributed estimations, based on their apparent complexity, using a numeric value attribution with a relative valuation, where they were estimated relative to each other, after deciding the most and least value to the most and least complex ones, respectively.

The *Sprint Review* and *Sprint Retrospective* meetings took place one day after each *Sprint*, with the supervisor, and the following *Sprints* were planned to begin as soon as possible.

In the end of each *Sprint*, a *Sprint Backlog* and a *Burndown Chart*[4] were available for the evaluation of the completed product (the tools and methods used for the gathering of this information are indicated and detailed in Section 4.2.7).

The *Code* artifact corresponds to the code of the initial *Throwaway Prototype* and the code of the *MasterVoicing* application. Their programming followed certain rules to facilitate the whole process and to produce code with the best quality possible, that were the following:

- All the files were organized and kept in a code repository (in a private project in *GitHub*);
- During programming, conventions for formatting, naming and commenting were followed, for better legibility and comprehension;
- Refactoring was done while developing, as an ongoing effort (some of this effort is noted in Section 4.2.7).

[2] Well defined unit of work, also known as a task, usually described with the template *"As a <type of user>, I want to <goal, objective>, so that I can <benefit, value>"* [138].

[3] Set of user stories with a similar theme [138].

[4] In the context of agile software development, it is a chart that shows how quickly user stories are closed/finished during a *Sprint* [139].

4.2.3 Initial Study

This section contains a description of the actions conducted to begin the achievement of the third and fourth objective steps described in Section 1.3, reiterated and labeled here to facilitate the train of thought:

- **Step 1 -** Familiarization and knowledge acquisition for the understanding of the iOS development environment and frameworks, with a special emphasis on the ones related to audio manipulation;

- **Step 2 -** Familiarization and knowledge acquisition regarding the programming languages involved in the development of the application and its algorithm (Objective-C, Swift and C++).

For the initiation of **Step 1**, an initial research was conducted, for a better understanding of the architecture of iOS, the available audio frameworks and the whole process of development of mobile applications for iOS (all briefly described in Chapter 3). The *Apple Developer* website [93] is a great resource for this matter, since it provides detailed informative guides, about all the tools and steps necessaries for the development and distribution of a mobile application for iOS, and several sample code projects [140].

Simultaneously, to start **Step 2**, it was also made a research about the available learning resources for Objective-C and Swift. The conclusions of this research were:

- A simple search in Google's search website [141] for "objective-c programming language" and "swift programming language" shows that there are many more results for Objective-C (as it would be expected from an older language) and that the Swift's top hits are much more recent than the ones about Objective-C (as it would be expected from a new language, that has become one of the most appreciated by developers [142]);

- It is easier to find updated tutorials and explained documentation for Swift - as opposed to tutorials about Objective-C that tend to be older and no longer supported for future changes and recent iOS versions [143] [144];

- The *Apple Developer* website promotes Swift and leaves Objective-C "in the background" - even thought the front pages promote Swift, many of the available guides and most of the sample code provided are still in Objective-C;

- The *Apple Developer* website has a lot of resources to learn Swift, that include documentation, sample code, videos and even *iTunes U*[5] courses [147];
- There are several online courses focused on iOS development [148] [149] [150] [151] [152] [153];
- There are dedicated blogs curated by developers about iOS programming in Swift [154] [155] [156];
- Despite being a relatively new language, there are already available various tools [157] and open source code (currently, a search for "language:Swift" in *GitHub*[6] results in over 35,000 repository results [159]) written by the developer community.

Instead of choosing the approach of "reading all the documentation thoroughly first", it was preferred a quick read of the most important guides and frameworks for the scope of the development of the application and to start "learning by doing". Therefore, taking two specific online courses, from *Udemy*[7] [161], seemed the most appropriate method to learn the fundamentals of iOS development. The courses were the following:

- "iOS 10 and Xcode 8 - Complete Swift 3 & Objective-C Course" [162];
- "The Complete iOS 10 & Swift 3 Developer Course" [163].

This two courses contributed to the continuation of **Step 2**, by providing a learning source about the basic syntax of the Swift and Objective-C languages, the process of designing the UI using the tools provided by *Xcode* and an insight about more advanced iOS features.

The final decision to make was choosing which IDE to use to initiate the learning process of Swift and Objective-C. Previous experience with *Xcode*, even thought only with C++ projects, from other course units, made easier the decision to use it. It is also the one recommended by Apple for the development to all of its devices, includes a set of useful development tools (summarized in Chapter 3) and facilitates the distribution process. Besides that, the chosen *Udemy* courses encourage its use and have sections explaining its basic functionalities.

After this initial phase, it was made a simple requirements analysis, explained in the following section, Section 4.2.4, and tests that would lead to the development of a prototype of the application (Section 4.2.5).

[5] "*iTunes U* is a dedicated section of Apple's *iTunes*" with several educational content from different sources, including well known learning institutions [145] [146].
[6] An online development plaftorm for code sharing and publishing, based on Git, with many added features [158].
[7] An online learning marketplace that provides courses taught by expert instructors [160].

It is important to note that all of the previous steps were just a means to start the learning process related to **Step 1** and **Step 2**, even thought this process was continuous, and continued until the final objectives' steps, with the development and distribution of *MasterVoicing*.

4.2.4 Initial Requirements Analysis

In every project of software development, it is important to make some kind of requirements analysis [130], that is usually done in the initial stages of the conception of the problem. This process, depending on the chosen development methodology, can be somewhat redone and the requirements may change or evolve over time.

After the initial study of the problem and some insight in the capabilities of iOS devices and its possible features, there was a general image of the fundamental behavior that the application could have. So, after that first step, an *Initial Requirements Meeting* took place, where the following basic functionalities were determined:

- The user should be able to use the application by whispering into the devices's microphone; after this, the device must output voiced speech processed from the inputted whispered voice;
- The user should be able to control whether or not the application is capturing his voice;
- The ideal situation is the one where there is no need for the use of headphones;
- The application should work in a "real-time" like manner; that is, the time interval between the point where the user speaks and the device outputs the processed result should be as small as possible.

These functionalities were studied in the initial tests (Section 4.2.5) and with the creation of the *Throwaway Prototype* (Section 4.2.6), and were later detailed, improved and reframed as *User Stories*, shown in Section 4.2.7, for the development of the application.

4.2.5 Initial Tests

The initial tests were based on different sample code projects, some sourced directly from the *Apple Developer* website and others from open source projects found on *GitHub*.

One of the projects was used to understand how to integrate C++ code into the application (for the encapsulation of the algorithm); the other ones were used to understand the audio capabilities and frameworks that could be used to deal with audio manipulation in real-time.

There are currently two main audio frameworks that can be used for audio programming in iOS, as seen in Figure 4.1: *AVFoundation* and *AudioToolbox*, that contain *AVAudioEngine* and *AUAudioUnit* (they provide different approaches to audio manipulation).

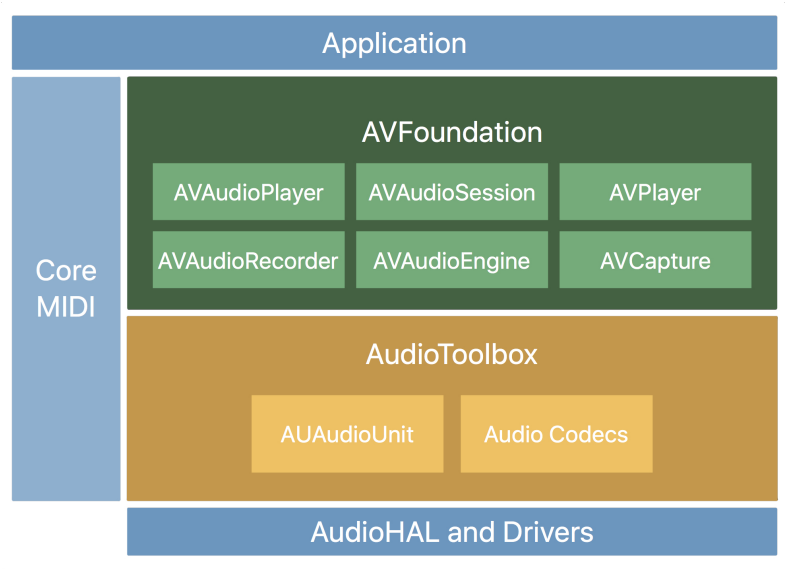


Figure 4.1: iOS audio stack (Adapted from [164])

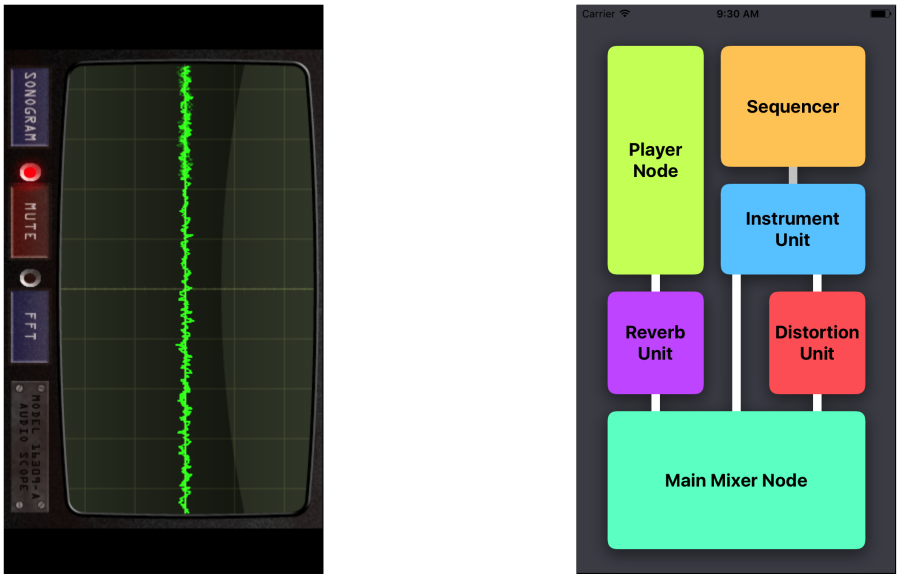
The first focus was on the study of the audio related projects, which can be divided in terms of programming language (Swift or Objective-C) and by the audio framework they use for audio input and output manipulation, as seen in Table 4.4.

Table 4.4: Sample projects - Capabilities and limitations

Project	Framework	Language	Capabilities	Limitations
aurioTouch	AudioToolbox	Objective-C	– Processes the audio input and output in real-time – Allows simultaneous recording and playback	– It is not intended to work with the built-in speaker – Only performs the modification of audio visually
aurioTouch2.0-Swift		Swift		
AVAEMixerSample	AVFoundation	Objective-C	– Processes the audio input and output in real-time – The processed audio is routed to the output	– Does not allow simultaneous recording and playback – Does not perform audio visualization
AVAEMixerSample-Swift		Swift		

These first two projects use the *AudioToolbox* framework, and are called *aurioTouch* [165] and *aurioTouch2.0-Swift* [166]. As the name suggests, the second project is an open source project that is a translation into Swift of Apple's *aurioTouch* sample code project (written in Objective-C). They are a demonstration of handling audio input and output using an *Audio Unit*[8] (particularly an *AURemoteIO*[9]) for the visualization of a regular time domain waveform (as shown in Figure 4.2a) and its correspondent frequency domain waveform (by computing a Fast Fourier Transform (FFT) of the input).

The other two projects, *AVAEMixerSample* [169] and *AVAEMixerSample-Swift* [170] (also a translated version into Swift of the original Apple's sample code project), offer a higher-level approach by using the *AVFoundation* framework. They demonstrate playback, recording and mixing using *AVAudioEngine*[10] (composed by different types of nodes with different capabilities, as shown in Figure 4.2b).



(a) *aurioTouch/aurioTouch2.0-Swift* (b) *AVAEMixerSample/AVAEMixerSample-Swift*

Figure 4.2: Sample projects - User interface [165] [166] [169] [170]

The study of the capabilities and limitations of these sample applications, stated in Table 4.4, were decisive for the choices made for the development of the *Throwaway Prototype*, explained in Section 4.2.6. Furthermore, to see if there were any advantages in terms of audio performance

[8] Audio manipulation and processing modules that can be used both inside an application or created as an extension to be used in a host app [167].

[9] An *AudioUnit* designed to handle simultaneous hardware input and output [168].

[10] Class that defines a set of connected audio nodes, that can be used to generate, process or mix audio [171].

between the usage of Swift or Objective-C, measures were made to the latencies of the audio session in each of the projects, and it was concluded that they were the same in both languages.

With this first sample projects, it was possible to test multiple features that allowed some room to understand which programming language and audio framework to use in the prototype, and also served as a basis for its development.

Finally, the sample project used for learn how to send information between Swift and Objective-C was: "SO-32541268 - Mix Swift, Objective-C, C and C++ files in the same *Xcode* project" [172]. This project provides simple "Hello World" methods that give an example of how to communicate between Swift, Objective-C, C and C++ files. With the basis of its code, it was possible to make a test application with two functionalities (Figure 4.3):

- A basic sum and multiplication calculator, that sends two inputted numbers from Swift to a C++ function, where it processes the calculations and returns them to Swift, to being showed in the UI;
- A more complex "Change Numbers" option, that involves the processing, in a C++ function, of an array of floats sent from a Swift file, returning to it with the processed numbers, and printing them into the console.

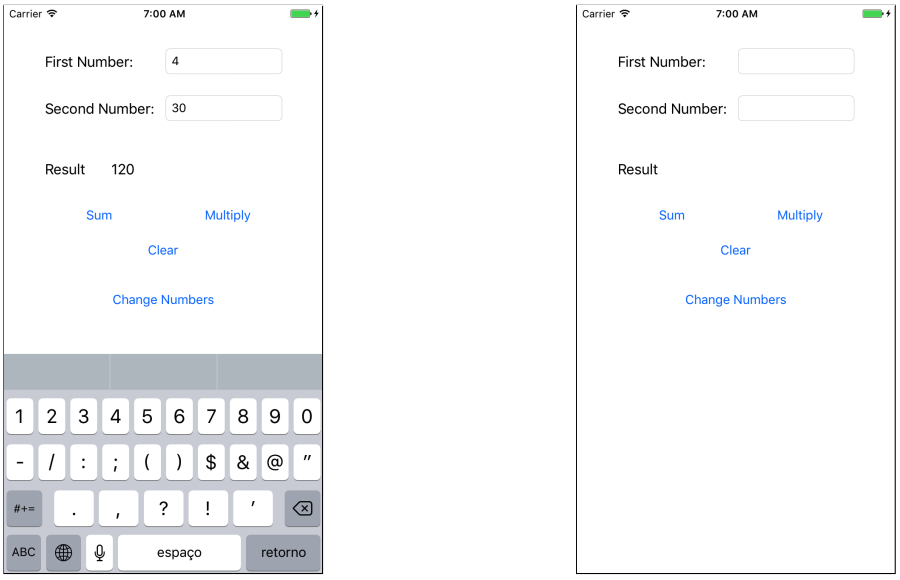


Figure 4.3: Test application adapted from *SO-32541268* [172] - User interface

The creation of this adapted test application involved the understanding of Objective-C wrapper files (files needed to connect Swift and Objective-C methods) and bridging header files (files needed to connect Objective-C and C++ methods).

After the confirmation that both of the wanted functionalities - audio related and algorithm encapsulation - were feasible by themselves, it was possible to move to the next step: the creation of a prototype that would confirm the possible capabilities of the application, by joining those two functionalities (Section 4.2.6).

In the end, the chosen applications to help with the creation of the prototype were the adapted version of *SO-32541268* and *AVAEMixerSample-Swift*; all of the other sample code was not necessary and was, therefore, discarded.

As a final note, it is also worth to mention that it was initially considered as a possibility the use of an external open source framework, such as *AudioKit* [173], but due to the lack of solutions for the real-time processing required and deprecation concerns, this idea was early abandoned.

4.2.6 Throwaway Prototype

With the experience provided by the initial tests, a set of decisions were made to initiate the development of a prototype of the application. The first main decisions were choosing which programming language and audio framework to work with. Since there were no actual differences between Objective-C and Swift noticed in the initial tests, and the initial research showed a more favorable recent documentation support for Swift, the most recent language was given priority for the development. The choice of the audio framework was the most challenging one, because of the known limitations of *AVFoundation* regarding audio processing. Nonetheless, it was the chosen one because it was thought it could fulfill the programming of the required functionalities (as recommended by Apple, higher level frameworks should be preferred over lower-level frameworks whenever possible [92]). It was also decided that, for testing the audio processing, an open source pitch shifting algorithm would be used [174], for the recreation of the complexity of the final voice processing algorithm. The final decision was the development approach: to choose whether to focus the prototype in the audio functionalities or the UI and usability workflows. Developing the main audio functionalities without worrying about the UI seemed the most appropriate approach (vertical prototyping approach). Summarizing, the characteristics of the developed prototype in terms of decisions are:

- **Programming languages:** always Swift when it is possible; used Objective-C for the bridging header between Swift and the algorithm in C++;
- **Audio framework:** *AVFoundation*;

- **Audio processing:** pitch shifting algorithm, that shifts the voice pitch higher or lower according to a hard-coded given factor;
- **Development approach:** creating a proof of concept with all the audio related problems solved, from the control of the audio session to solving the problem of audio processing with a C++ algorithm.

The architecture logic of the audio engine, implemented in the prototype, is depicted in Figure 4.4. The audio is captured from the devices' microphone by the *Input Node*, where it is installed a tap, to process the audio and to schedule it to play in the *Real Time Player*. The *Real Time Player* routes the audio through its connection with the *Main Mixer Node*, where a tap is also installed for recording the audio in a *File*. The *Recording Player* routes the recorded audio, contained by the *File*, when required, back to the *Main Mixer Node*. The *Main Mixer Node* is connected to the *Output Node*, that provides that the audio it receives is sent to the chosen output (built-in speaker, internal hear speaker, external output).

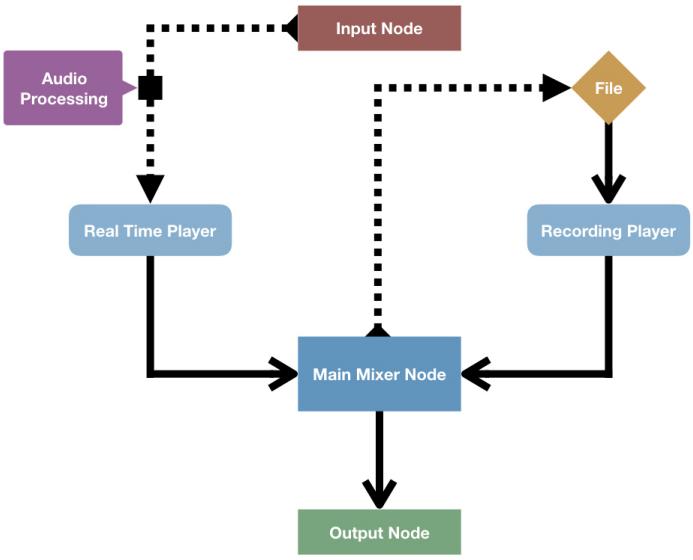


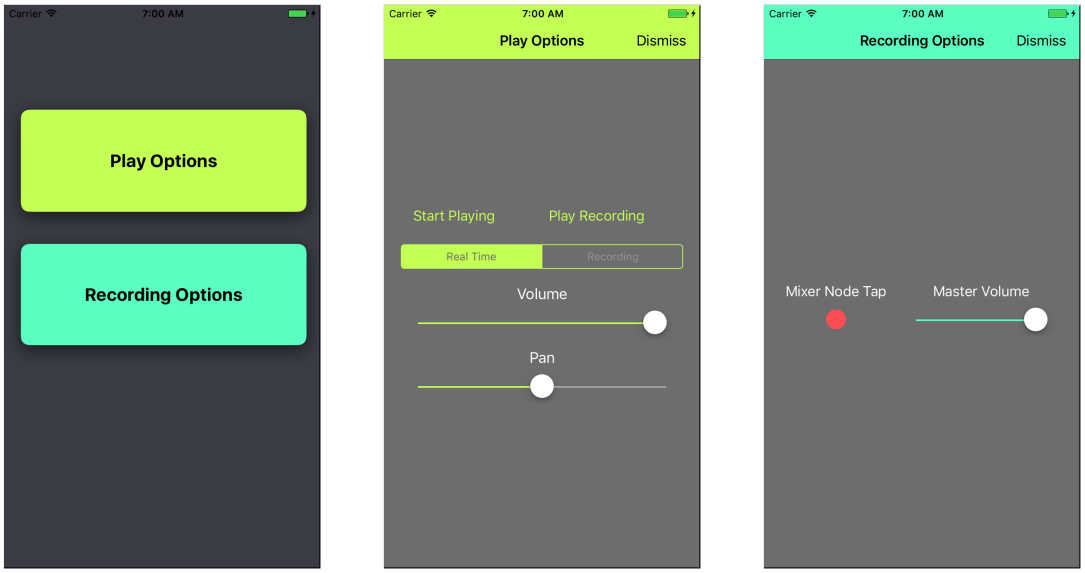
Figure 4.4: *Throwaway Prototype* - Audio engine architecture (Created with [175])

The understanding of this architecture was time demanding and rather challenging, because *AVAudioEngine* is not usually used or firstly designed for the mix of real-time audio manipulation and processing.

The final features of this prototype, whose UI can be seen in Figure 4.5, were the following:

- Capturing audio in real-time from the device's internal microphones;
- Processing and playback of the captured audio in real-time;

- Recording the processed audio in a temporary file for later playback;
- Playback of the recorded audio file;
- Changing recording and playback volumes and playback panning.



(a) Main menu (b) Play options menu (c) Recording options menu

Figure 4.5: *Throwaway Prototype* - User interface

The main purpose of the creation of this prototype was to confirm the possibility of development of the desired features of *MasterVoicing* and to better understand the audio engine and audio session management, by using the sample code related with those functionalities as a template. However, at its development, several problems were encountered anyway, too specific and numerous to be accounted for in this document, mostly related with the correct audio manipulation, that were successfully overcomed.

4.2.7 Application Development

The application was developed using the previous described (Section 4.2.2.3) software development methodology. The development was divided in three *Sprints*, described in Sections 4.2.7.2, 4.2.7.3 and 4.2.7.4, whose themes were:

- **Sprint 1 -** Main Features;
- **Sprint 2 -** Advanced Features;
- **Sprint 3 -** Final Improvements.

Before the development process started, from the 28th of June to the 14th of July, a set of steps were made for its preparation; these are described in Section 4.2.7.1.

A visual timeline of all the involved steps of the development can be seen in Table 4.5; these events were previously explained in Section 4.2.2.3.

Table 4.5: Development tasks - Timeline

Months	Jun	Jul						Aug					Sep
Tasks / Days	28	15	16	26	27	28	29	10	11	27	28	30	3
Preparation	■												
Sprint 1 Planning		■											
Sprint 1			■										
Sprint 1 Review				■									
Sprint 1 Retrospective					■								
Sprint 2 Planning						■							
Sprint 2							■						
Sprint 2 Review								■					
Sprint 2 Retrospective									■				
Sprint 3 Planning										■			
Sprint 3											■		
Sprint 3 Meeting												■	
Sprint 3 Review													■

■ *Completed*

4.2.7.1 Development Preparation

Before starting the development of the application, some steps needed to be addressed:

- Outline the adapted software development methodology to use;
- Choose the appropriate tools to put the methodology in practice;
- Prepare the necessary environment and install the necessary tools;
- Delineate the desired features and general appearance of the application.

This means that, firstly, it was necessary to make a thorough research about current development methodologies, in order to choose one and adapt it for the current work (Section 4.2.2).

Following this decision, it was made a research about the possible tools to use to help to put the methodology in practice; the chosen ones were *GitHub* and *ZenHub* [176]. *ZenHub* is directly connected with *GitHub*, it is free for single developer use and it is very easy to install, which made it the most appealing choice.

After creating a code repository in *GitHub* and installing *ZenHub*, the next step was to think about the desired features of the application and create *User Stories* to express them (enumerated in Tables 4.6, 4.7 and 4.8). At the same time, it was necessary to think about the appearance of the application, that is, the UI. For this purpose, some research was made, with a main focus in real applications' UIs and the official Apple recommendations (principally the Apple "Human Interface Guidelines" for iOS [177]). With this information, some mockups of an UI were made, which final version can be seen in Figure 4.6. The mockups were made using *Figma*, a "collaborative interface design tool" [178], using UI elements from "iOS Design Kit" [179].

Figure 4.6: *MasterVoicing* user interface mockups (Created with [178])

The final steps of the preparation were to learn how to use *ZenHub*[11], add the previously created *User Stories* to its interface, estimate the effort of each one with *Story Points*[12], and plan and create *Epics* and *Sprints*.

[11] For clarification purposes, in *GitHub* and *ZenHub*, *Sprints*, *User Stories* and *Story Points* have different designations, respectively: *Milestones*, *Issues* and *Estimates* [180].

[12] The *Story Points* were estimated using the default *ZenHub*'s estimation method - a set of fibonacci numbers, with an extra maximum value of 40: 1, 2, 3, 5, 8, 13, 21 and 40.

4.2.7.2 *Sprint 1*

For the first *Sprint*, *Epics* were chosen with the purpose to create an initially basic, but working, version of the application. This *Epics* and its *User Stories* are detailed in Table 4.6, which also contains the completion date of each one, which evaluation, regarding the correspondent story points, result in a burndown chart, depicted in Figure 4.7.

Figure 4.7: *Sprint 1* - Burndown chart (Adapted from *ZenHub* [176])

It was also decided that 10 days would be an ideal duration for the first *Sprint* (the total estimation of its *Story Points* were 104 points, which resulted in a reasonable average of 10 points per day), and the next ones would be adapted from its completion velocity.

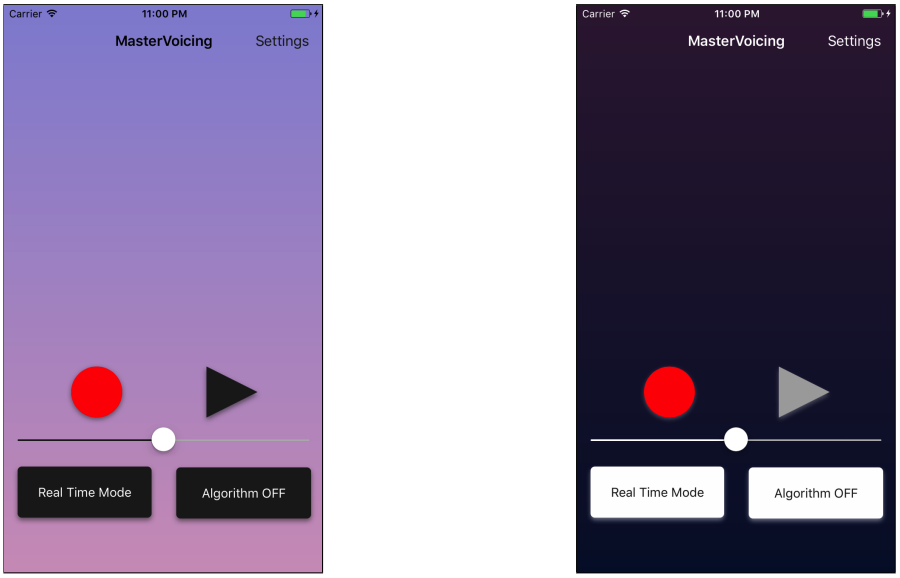
The most challenging features, noted at the *Sprint Review*, were, as expected, the ones of the "Basic User Interface" *Epic*. There was also a bug at the end of the *Sprint*, while recording audio, where sometimes the application would crash after retrying to record, only with the voice processing turned on, that was resolved in *Sprint 2*, by rearranging the audio engine architecture and logic. After this first *Sprint*, the audio engine architecture was equivalent at the *Throwaway Prototype* one (Figure 4.4), since it was the basis for many of the features of this *Sprint*. Another thing noted in the *Sprint Review* was that, by looking at the burndown chart, it was clear that the first two big user stories should have been more specific to begin with, which was why it was decided, in the *Sprint Retrospective*, that the next *Sprint*'s ones should be reviewed and made smaller or more detailed, which is noticeable in its burndown chart, in Figure 4.9.

Table 4.6: *Sprint 1* - User stories

ID	Title	SP	Description	CD	Epic
US01	Basic UI Design	40	As a user, I want to have visual and easy to use controls, so that I can interact with the application and understand what I can do in terms of features.	Jul 23	Basic User Interface
US02	Basic UI Structure	40	As a user, I want that the application works logically in the expected way and responds to the visual commands, so that I can use it without problems.		
US03	Record Voice	5	As a user, I want to record my voice, so that I can use it with other features of the application.	Jul 25	Basic Audio Processing
US04	Playback Voice	5	As a user, I want to playback a processed version of my recorded voice, so that I can be heard and understood.		
US05	Change Volume	1	As a user, I want to change the volume of the recording, so that I can adapt it to my needs.		
US06	Playback Output: built-in speakers	2	As a user, I want the playback to work with the built-in speakers of my device, so that I can use it everywhere without extra accessories.		
US07	Audio Processing: algorithm adaptation and encapsulation	3	As a user, I want that the application uses an algorithm for audio processing, so that I can hear my voice being altered in real time or after recording it.		
US08	Playback Output: headphones	5	As a user, I want the playback to work with headphones, so that I have more privacy while using the application.		More Playback Options
US09	Playback Output: external speakers	3	As a user, I want the playback to work with external speakers, so that I can use better quality accessories for audio output.		

SP - Story Points **CD -** Completion Date

The first UI implementation was inspired by the design of the previously shown mockups, and the final version of the UI after *Sprint 1* can be seen in Figure 4.8. The reason why its basic implementation took so long, even thought there was a simple way of using the native UI elements of iOS, was that it was decided to custom-draw or change all the UI elements, and create custom-made animations to make them more appealing. This is more detailed in Section 4.4.



(a) Light colors template (b) Dark colors template

Figure 4.8: *Sprint 1* - User interface

4.2.7.3 *Sprint 2*

The second *Sprint* focused on the improvement of the current features and on the addition of advanced features, described in Table 4.7.

By evaluating the burndown report of *Sprint 1*, in the *Sprint 1 Review*, it was possible to plan the duration of *Sprint 2*. According to its estimated 129 *Story Points*, at the same velocity as *Sprint 1*, *Sprint 2* was planned to have the duration of 12 days.

The burndown chart of this *Sprint* (Figure 4.9) shows that the choice, made in the previous *Sprint Retrospective*, of making the *User Stories* more specific and detailed, resulted in a more smooth development.

At the end of *Sprint 2*, a new bug remained to be solved, related with the handling and changing the audio output. The bug produced an unexpected behavior in the application when the output changed while processing the audio, that is, in real-time mode, while recording or while playing the previous recorded audio. This bug was handled in *Sprint 3*, with *User Story US27*.

Table 4.7: *Sprint 2* - User stories

ID	Title	SP	Description	CD	Epic
US10	Pause Playing	13	As a user, I want to pause the playback of a recording, so that I can control the playback while it is not finished.	Jul 31	Basic Settings and Audio Processing
US11	Basic Settings Structure and UI	13	As a user, I want to have a list of settings, so that I have access to more options.	Aug 6	
US12	Settings: max. recording duration	13	As a user, I want to change the maximum recording duration, so that I have better control on the internal audio files size.	Aug 7	
US13	Settings: activate/deactivate voice processing	5	As a user, I want to be able to activate/deactivate the processing of my voice, so that I can choose how to record it.	Jul 30	
US14	Show Time: recording	5	As a user, I want to see the audio recording time, so that I know the remaining and elapsed time.	Jul 31	Extra UI Features
US15	Show Time: recorded file playback	5	As a user, I want to see the playback time of the audio file that is playing, so that I have a visual feedback of the playback.		
US16	Audio Visualization	40	As a user, I want to have some visual feedback of the audio, so that I know if it is being recorded or played.	Aug 2	
US17	UI: visual improvements and feedback	13	As a user, I want that the visual interface of the application is pleasant to the eye and provide visual feedback of its state, so that its usage is more enjoyable and useful.	Aug 9	
US18	Settings: shake to change processing mode	1	As a user, I want to switch the feature of changing the processing mode with motion, so that I don't use it unintentionally.	Jul 30	Extra Settings
US19	Settings: choose color theme	8	As a user, I want to change the color theme, so that I can personalize it and adapt it to current light conditions.	Aug 6	
US20	Settings: change language	13	As a user, I want to change to a language more familiar to me, so that I better understand the features.		

SP - Story Points **CD -** Completion Date

Figure 4.9: *Sprint 2* - Burndown chart (Adapted from *ZenHub* [176])

The main challenge of *Sprint 2* was the implementation of the audio visualization. After a general research about pre-made available solutions for audio visualization, it was concluded that none of them would be adaptable to an audio engine with the same configuration of the one implemented in *Sprint 1* and the particular aspect of using the scheduling of audio buffers to a player node (better explained in Section 4.4). Therefore, it was decided to create a custom audio visualization, which meant that the configuration of the audio engine needed to be altered. This new configuration, that adds two mixer nodes for the visualization is shown in Figure 4.10.

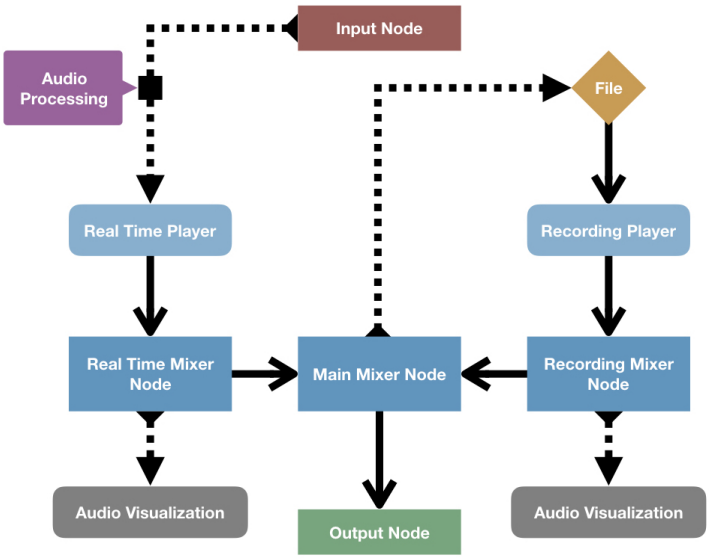


Figure 4.10: *Sprint 2* - Audio engine architecture - After implementing the audio visualization (Created with [175])

However, to solve the previous bug in recording, it was decided to reconfigure the audio engine architecture, which resulted in what can be seen in Figure 4.11.

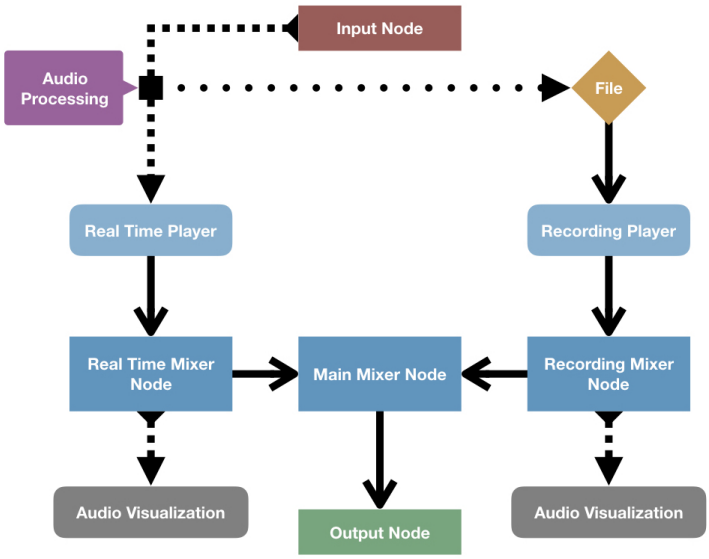


Figure 4.11: *Sprint 2* - Audio engine architecture - After refactoring (Created with [175])

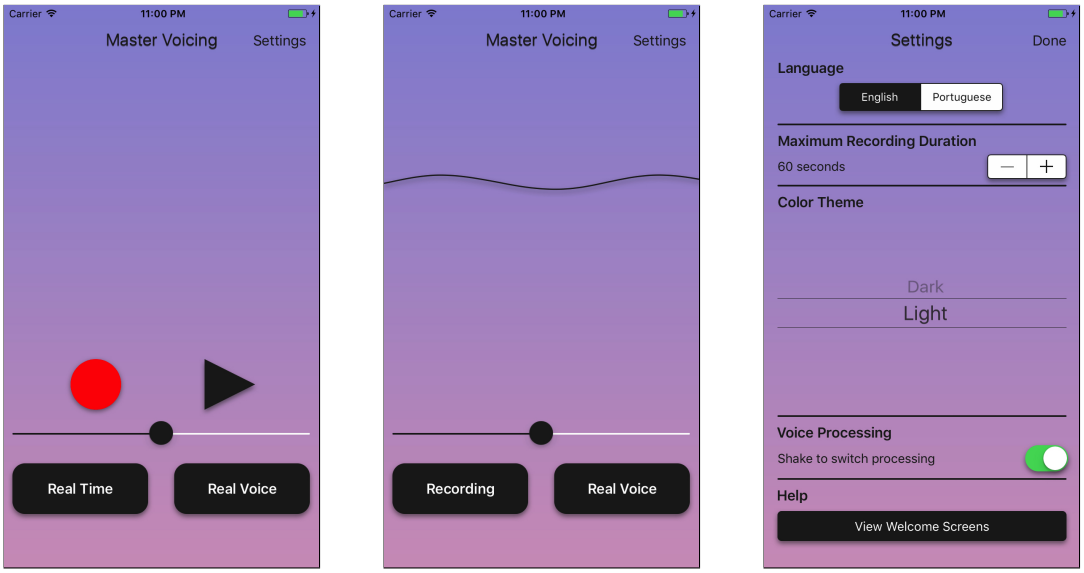
The implementation of the audio visualization requires the installation of a "tap" in a node of the audio engine (typically a mixer node) that receives the audio buffers from the connected nodes. The first architecture, used in the prototype, wasn't thought to include audio visualization, so, it only included the *Main Mixer Node*, whose tap was already being used for audio recording purposes (it is not possible to install two taps simultaneously in the same node, and it is not viable to capture, process, record and use the same buffer for visualization, as it may cause audible breaks in the audio playback). The new configuration allows to install a tap in each one of the added mixers (*Real Time Mixer Node* and *Recording Mixer Node*), and to use them when necessary.

One suggested and thought out feature which development was started but abandoned, because of time constraints, as decided in the *Sprint 2 Retrospective*, was the option of sharing a recorded audio file (with the added possibility of changing the name of the file before sharing), which UI components can be seen in Figure 4.12.

Figure 4.12: *Sprint 2* - Abandoned feature UI components - Edit name and share audio file

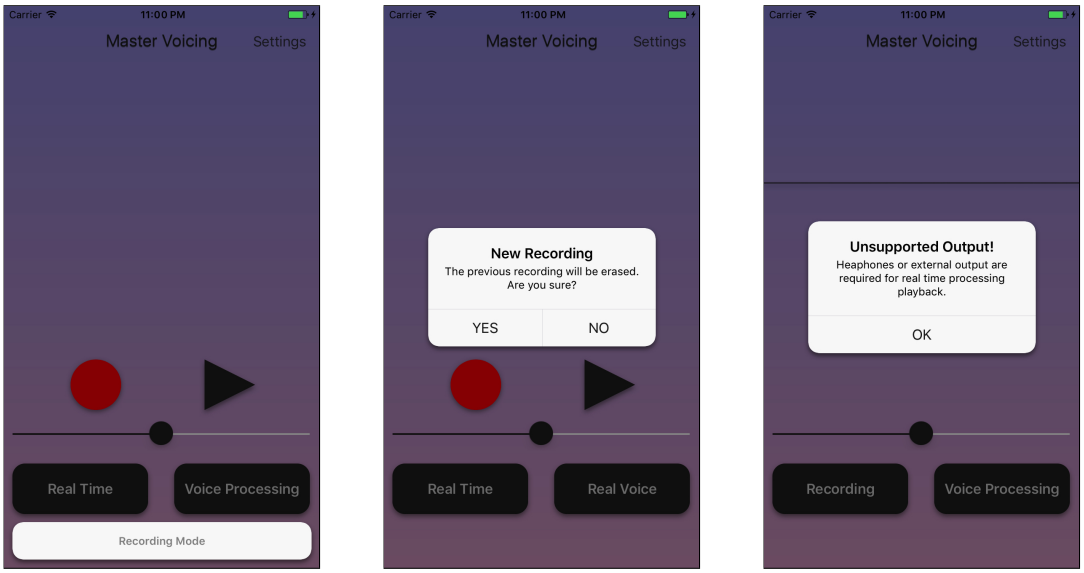
Also in the *Sprint 2 Retrospective*, it was suggested by the *Product Owner* that a new UI element was added, as a new feature for the final *Sprint 3*. This element would be a control for a given changeable characteristic of the algorithm; as for the pitch shifting algorithm, this means controlling the pitch factor (higher or lower than the real voice pitch).

Many of the implemented features in *Sprint 2* translated in many improvements in UI design and visual feedback, being some of them depicted in Figures 4.13 and 4.14.



(a) Recording mode UI (b) Real time mode UI (c) Settings UI

Figure 4.13: *Sprint 2* - User interface - Implemented features



(a) Processing mode changes alert (b) Confirmation alert (c) Unsupported output alert

Figure 4.14: *Sprint 2* - User interface - Visual feedback

After a last refactoring of the code, it was possible to simplify even more the audio engine architecture and eliminate the two specifically created audio mixer nodes for visualization, as shown in Figure 4.15. This was possible by using a single tap in the *Main Mixer Node*, that was no longer tapped, thanks to the new audio configuration for audio recording to a *File*.

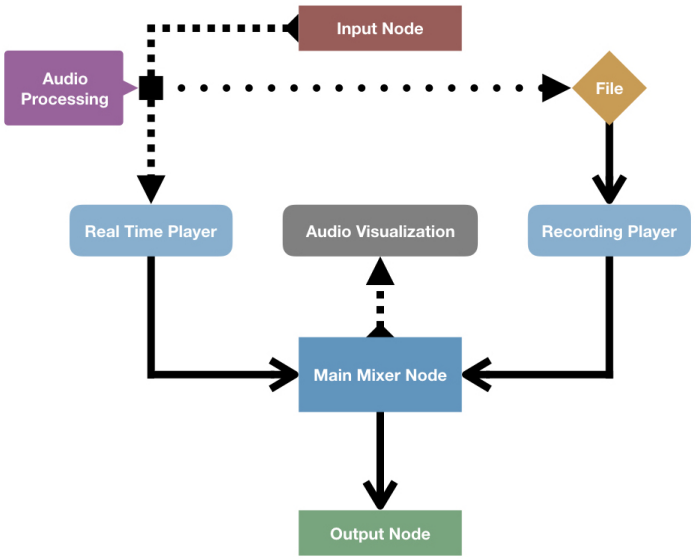


Figure 4.15: *Sprint 2* - Audio engine architecture - Final version (Created with [175])

4.2.7.4 *Sprint 3*

The third and final *Sprint* consisted in the development of final improvements, described by the *User Stories* in Table 4.8, whose burndown chart is shown in Figure 4.16.

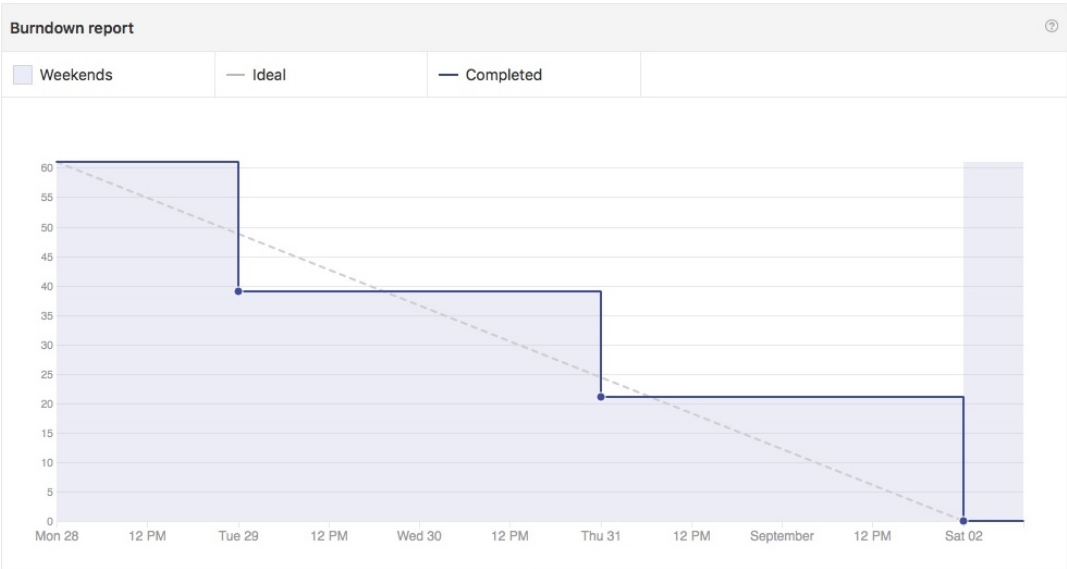


Figure 4.16: *Sprint 3* - Burndown chart (Adapted from *ZenHub* [176])

Table 4.8: *Sprint 3* - User stories

ID	Title	SP	Description	CD	Epic
US21	Algorithm's Property Alteration Slider	13	As a user, I want to have an UI element, such as a control slider, so that I can easily alter, in real time, a property of the algorithm.	Aug 29	—
US22	More Color Themes	3	As a user, I want to have a set of different color themes to chose, so that I can have a more personalized user experience.		Final UI Features
US23	UI Element: "Done" Button	3	As a user, I want to have a clearly identifiable "Done" button, so that I can better understand its meaning.		
US24	UI Element: "Settings" Button	3	As a user, I want to have a clearly identifiable "Settings" button, so that I can better understand its meaning.		
US25	Help: view application's instructions	13	As a user, I want the application to have a visual aid that explains its instructions, so that I can better learn how to use its different functionalities.	Aug 31	
US26	Application Icon Design	5	As a user, I want the application to have a distinct and identifiable icon, so that I can clearly distinguish it from the other applications on my device.		
US27	Support Changing Audio Routes	21	As a user, I want that the application supports changing to a different audio output, so that I can use the output of my choice.	Sep 2	—

SP - Story Points **CD -** Completion Date

This *Sprint* had the duration of 6 days, calculated according to the effort estimation (61 *Story Points*) that was made in the *Sprint 3 Planning*, the day after the final definition and estimation of the remaining *User Stories*, on August 27th. During the *Sprint*, in a meeting with the *Product Owner*, on August 30th, it was decided that there would be no possibility of having a well enough developed whisper-to-speech algorithm, due to time constraints. Therefore, the final version of the application uses the previous pitch shifting algorithm used in the *Throwaway Prototype*, described in Section 4.2.6. Further information about this algorithm and its integration with the application is given in the next section, Section 4.3.

In *Sprint 2*, the *User Story US25* was thought to be presented differently than what was implemented in *Sprint 3*. As previously shown in Figure 4.13c, the instructions of the application were meant to be seen at the launch of the application and later accessible to the user in the settings' menu. In the final version of the application, this feature is presented in the main screen as a helping feature, by using an overlay with its instructions.

The most challenging parts of this *Sprint* were the implementation of the instructions and the support for handling audio route changes. The first part required creating a more complex management of the UI views and correspondent view controllers and the second part required a better understanding of the audio engine and session management.

The additions and improvements made to the UI in *Sprint 3* resulted in the final version of the *MasterVoicing* UI, shown in Section 4.4.

In the *Sprint 3 Review* it was concluded that all the desired functionalities for having a good enough final product were achieved, since it would likely be suited for submission through the *App Store*, if the initially thought whisper-to-speech algorithm had been the one used.

Since this last *Sprint* concluded the implementation of the final version of the application for this work's scope, there was not a *Sprint 3 Retrospective* after the *Sprint 3 Review*.

4.3 Algorithm

This chapter contains a simple explanation about the algorithm included in the *MasterVoicing* application and its implementation. The algorithm is the pitch shifting algorithm mentioned in Section 4.2.6, that was used as a proof-of-concept in the creation of the *Throwaway Prototype* and for test purposes during the development of the final product. This algorithm uses a STFT (Short Time Fourier Transform) "for changing the perceived pitch of an audio signal by representing it as a sum of sinusoids and scaling the frequency of these sinusoids" [181].

4.3.1 Implementation

The algorithm is implemented using a C++ class and an Objective-C wrapper class, that connects the C++ class to Swift. The whole process involves five different files, correspondent to three different classes: *Algorithm.hpp*, *Algorithm.cpp*, *Algorithm-Wrapper.h*, *Algorithm-Wrapper.mm* and *AudioEngine.swift*. In *AudioEngine.swift*, the *installInputNodeTap* method (Listing 4.1) calls the wrapping function of the Objective-C wrapper class (Listing 4.2), which is directly connected

with the C++ class that contains the algorithm. The C++ class (Listing 4.3) contains a set of functions that are responsible for the audio processing of an incoming buffer.

```
private func installInputNodeTap()
{
    if(_isInputNodeTapped == false)
    {
        _engine.inputNode?.installTap(onBus: 0, bufferSize: self._numOfSamples,
            format: self._playerFormat) {buffer, when in
            buffer.frameLength = AVAudioFrameCount(AudioProperties.sampleSize)

            if(self._algorithmOn == true)
            {
                // call the algorithm's wrapping function
                Algorithm_Wrapper().pitchShiftAlgorithm_wrapped(buffer.
                    floatChannelData?.pointee, andPitch: self._algorithmProperty)
            }

            if(self._isRecording == true)
            {
                do {
                    try self._mixerOutputFile.write(from: buffer)
                } catch let error  {
                    fatalError("error writing buffer data to file, \(error.
                        localizedDescription)")
                }
            }

            if(self._scheduleInputBuffers == true)
            {
                // schedule player to play the buffers in sequence
                self._player.scheduleBuffer(buffer, completionHandler: nil)
            }
        }
        _isInputNodeTapped = true
    }
}
```

Listing 4.1: Swift function: *installInputNodeTap()*

```
- (void)pitchShiftAlgorithm_wrapped:(float *)buffer andPitch:(float)value
{
    Algorithm().pitchShiftAlgorithm(buffer, value);
}
```

Listing 4.2: Objective-C wrapper function to call the algorithm in C++

```
class Algorithm {
public:
    Algorithm();
    ~Algorithm();
    void pitchShiftAlgorithm(float * buffer, float pitchValue);
    void smbPitchShift(float pitchShift, long numSampsToProcess, long fftFrameSize,
         long osamp, float sampleRate, float *indata, float *outdata);
    void smbFft(float *fftBuffer, long fftFrameSize, long sign);
    double smbAtan2(double x, double y);
};
```

Listing 4.3: *Algorithm* class header code of *Algorithm.hpp* [174]

Finally, the function *pitchShiftAlgorithm* calls the *smbPitchShift* function (Listing 4.4), which is the basis function of the algorithm. They both receive a *pitchShift* factor value between 0.5 (one octave down) and 2 (one octave up), where 1 does not alter the pitch in any way [181]. This factor is controlled by the *Pitch Slider* in the UI of the application.

```
1 void Algorithm::pitchShiftAlgorithm(float * buffer, float pitchValue)
2 {
3     smbPitchShift(pitchValue, BUFFER_SIZE, MAX_FRAME_LENGTH, 4, SAMPLE_RATE, buffer
          , buffer);
4 }
```

Listing 4.4: C++ function that calls the *smbPitchShift* function

Besides the pitch changing factor, there are other required parameters that have to be filled to process the audio [181]:

- *numSampsToProcess* - number of buffer samples;
- *fftFrameSize* - FFT frame size used for the audio processing;
- *osamp* - oversampling value;
- *sampleRate* - audio's sample rate;
- *indata* - input buffer before processing;
- *outdata* - output buffer after processing.

In the final product, the values of *BUFFER_SIZE* and *MAX_FRAME_LENGTH* correspond to 512 and *SAMPLE_RATE* equals 22050.0 Hz, as these were the required values for the intended whisper-to-speech algorithm. The oversampling value was tested to be used at its maximum value (32), which would work without any problems, but it was left at the minimum possible value (4), because it provided an apparent slightly decrease in audio processing latency.

One of the main advantages of this algorithm is that the input audio buffer can be the same as the output buffer, which means the data can be processed in-place, without memory allocations, that usually provoke audio playback problems, such as silent breaks between different buffer's playback. Further details about this algorithm are better explained in a related article [181] and can be better understood with its source code [174].

After this whole process, that is repeated for each audio buffer, the altered audio buffer is scheduled to play sequentially in a player (Listing 4.1), that redirects it to the main mixer node, so that the audio can be heard in real time through the output node.

4.4 Final Product

This chapter documents some details about the features, UI and the developed code structure of the *MasterVoicing* application - the final product of this work.

4.4.1 Features

The application's features can be divided in two categories: *Main Features* and *Extra/Advanced Features* (Table 4.9). The former are the most important, because the application wouldn't make sense without them, whereas the latter are just enhancements to the original desired functionalities.

Table 4.9: *MasterVoicing*'s features

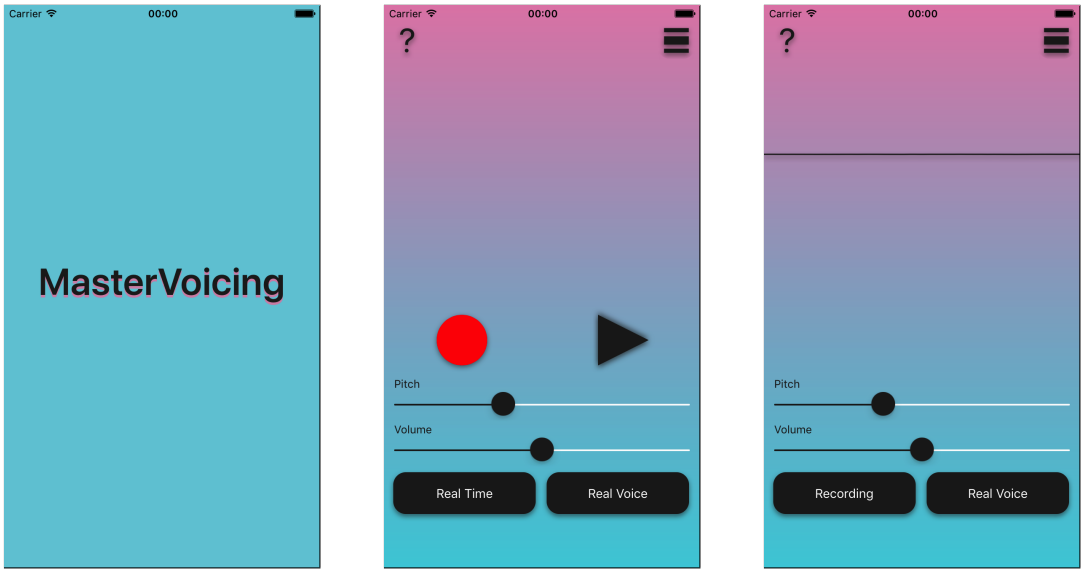
Category	List
Main Features	• Two modes for voice recording: – *Real Time* and *Recording*. • Two audio processing modes: – *Voice Processing* and *Real Voice*. • The capability of choosing the desired pitch of the audio processing and the general volume of the audio output.
Extra/Advanced Features	• Choose the application's language, between English or Portuguese. • Choose the maximum recording duration, while in recording mode. • Choose a color theme from a set of predefined color themes. • Change the voice processing mode, by shaking the device, and having the capability of switching this functionality on or off. • Instructions that aid the understanding of the application's features.

The UI of these features are shown in the next section, Section 4.4.2, and the structure of its developed code are explained in Section 4.4.3.

4.4.2 User Interface

Throughout the design of the UI of the application, the main focus was on the simplicity of the UI elements and the ease of user interaction.

When the application is first launched, the user is presented with a launch screen, as shown in Figure 4.17a, before the initial screen of the application appears. At its initial state, the application is presented in the *Recording* mode (playback mode) and in the *Voice Processing* mode (audio processing mode), as seen in Figure 4.17b. The language used in the UI elements is the device's current language and the used colors are the ones of the *Default* color theme.

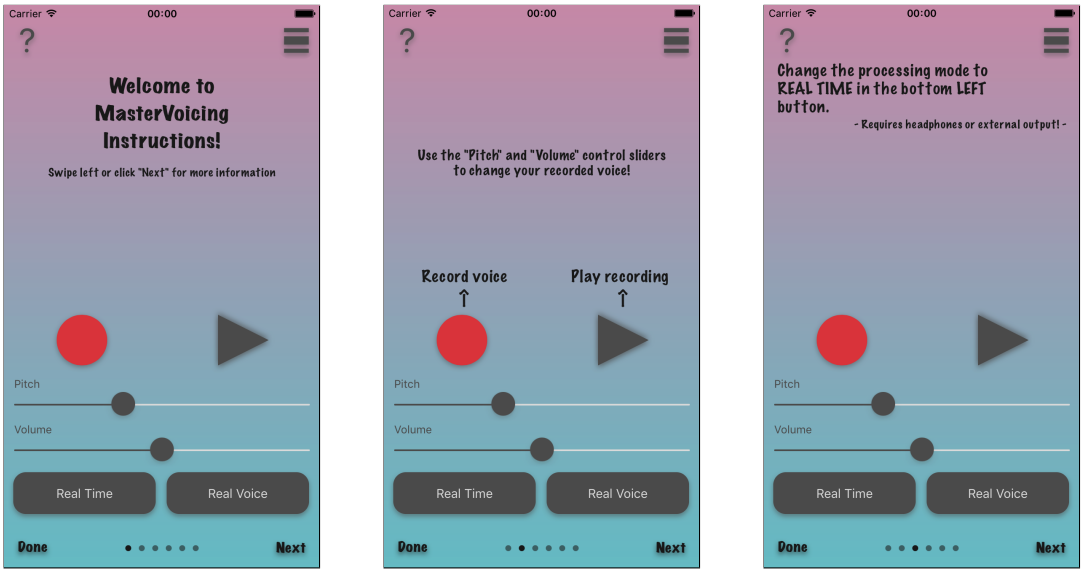


(a) Launch screen UI (b) Recording mode UI (c) Real time mode UI

Figure 4.17: *MasterVoicing*'s user interface - Main features

The user can change the playback mode to *Real Time* mode (Figure 4.17c), in the respective button. In *Real Time* mode, the application records, processes and plays back the audio without the need of any user assistance. On the contrary, in *Recording* mode (Figure 4.17b) the user has to go through the process of interacting with the UI for the recording and playback of the audio. In both modes, the user needs to interact with the UI for choosing the desired pitch and volume values or the audio processing mode. The user can switch the audio processing on or off by choosing, respectively, the *Voice Processing* or *Real Voice* mode buttons.

Since the *Main Features* of the application are related with the recording, processing and playback of voice, its UI was thought out to be designed around that purpose. For that reason, the *Extra Features* are not the first thing the user sees on the screen; each one of them has a button to access them, more specifically, the *Help* (Figure 4.18) and *Settings* (Figure 4.19) buttons.



(a) Instructions UI - First page (b) Instructions UI - Second page (c) Instructions UI - Third page

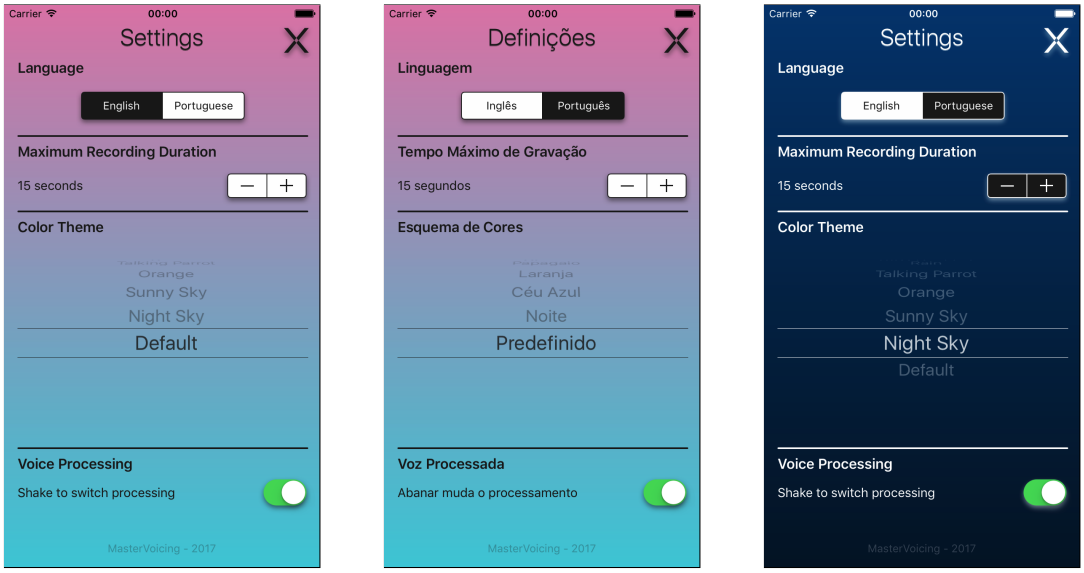
(d) Instructions UI - Fourth page (e) Instructions UI - Fifth page (f) Instructions UI - Sixth page

Figure 4.18: *MasterVoicing*'s user interface - Instructions

The design of the *Instructions* was inspired by one of the usual methods for helping users to understand UI features: instructional overlays [182]. They are available in both of the supported languages and the user can navigate them by swiping left or right, or clicking the *Next* button. After the user is ready to go back to using the application, he/she has to click on the *Done* button.

In *Settings*, the user has access to the rest of the *Extra Features*, which are described in the previous section (Section 4.4.1). One of these features involves choosing the application's color theme, from 13 different predefined color themes, whose colors and names were inspired by a

set of images gathered by searching the name of the themes in *Google* (with the exception of the *Default* theme, which was inspired by the resulting images of searching the terms "Pink and Blue Candy"). The colors were picked from the images using an online tool that helps generate color schemes - *Coolors* [183]. An overall view of those themes can be seen in Figure 4.20, ordered by its respective name, from left to right: *Dark*, *Light*, *Purple Flower*, *Watermelon*, *Purple Skin*, *Beach*, *B&G Cocktail*, *Rain*, *Talking Parrot*, *Orange*, *Sunny Sky*, *Night Sky* and *Default*.



(a) Predefined UI (b) UI in Portuguese (c) UI with *Night Sky* color theme

Figure 4.19: *MasterVoicing*'s user interface - Settings

Figure 4.20: *MasterVoicing*'s user interface - Color themes (Created with [184])

Besides these features, the application contains a set of custom animations and alerts (implemented in *Sprint 2* and previously shown in Figure 4.14) whose purpose is to give the user some feedback about changes and different states of the application (Tables 4.10 and 4.11).

Table 4.10: *MasterVoicing*'s user interface - Custom animations

Animation	Trigger	Description
Recording button	– When the recording starts	– The recording button is animated with size and opacity changes, that last until the recording stops
Play to Stop button	– When the recording starts	– The play button is animated to be transformed into the stop button
Stop to Play button	– When the recording is stopped	– The stop button is animated to be transformed into the play button
Play to Pause button	– When the playback starts or is resumed	– The play button is animated to fade away and the pause button is animated to fade in on its place
Pause to Play button	– When the playback is paused or finished	– The pause button is animated to fade away and the play button is animated to fade in on its place
Play button emphasis	– When there is no audio recording to play	– The play button is animated with a "shaking" effect
Stop button emphasis	– When the record button is pressed after the recording is already in progress	– The size of the stop button is momentarily enlarged to remind the user that it is already recording
Audio modes buttons	– When the audio playback or audio processing changing modes buttons are pressed	– The buttons are animated with a reversible position displacement
Audio visualization	– When the audio is being recorded or played, both in recording mode and real time mode	– A representation of the audio being recorded or played is presented

Table 4.11: *MasterVoicing*'s user interface - Alerts

Alert	Trigger	Description
Playback Mode Changes	– When the user presses the audio playback mode changing buttons	– The alert appears momentarily in the bottom of the screen with the name of the currently chosen audio playback mode
Processing Mode Changes	– When the user presses the audio processing mode changing buttons	– The alert appears momentarily in the bottom of the screen with the name of the currently chosen audio processing mode
New Recording	– When the user attempts to start a new recording and a previous recording was already made	– The alert informs the user that a new recording will overwrite the current one, and asks to confirm the start of the new recording
Unsupported Output	– When the audio output is the built-in speaker of the device and the user changes to *Real Time* mode	– The alert informs the user that the current output is unsupported and informs which are the supported ones

To avoid problems regarding low image quality with the resizing of the images, none of the UI elements uses images for its representation. Instead, they are all custom made or adapted from the native UI elements available. The following UI elements of the application are all custom designed and drawn in the views by using a drawing method that uses bezier curves: *Record* button, *Play* button, *Stop* button, *Pause* button, *Settings* button, *Done* button. The other UI elements are all adapted to match the custom appearance of the UI. Other important aspect of the UI is the custom developed audio visualization, that is also a custom made drawing of the audio buffer values, adapted to the devices' screen size, using the same drawing method as the custom-made UI elements. The final design aspect of the application is its Icon, seen in Figure 4.21.

Figure 4.21: *MasterVoicing*'s icon (Created with [184])

4.4.3 Code Structure

The following Tables 4.12 and 4.13 contain a description of all the Swift classes of the developed code of the application, and their associated views.

Table 4.12: *MasterVoicing*'s developed code - Classes (Swift)

Name	Description	Associated View
AppDelegate	– The application's delegate	—
ViewController	– The application's main view controller	*Main View*
SettingsViewController	– View controller class	*Settings View*
InstructionsViewController	– View controller class	*Instructions View*
InstructionsPageViewController	– View controller class	—
InstructionsGeneralViewController	– View controller class	—
FirstInstructionsPageViewController	– View controller class	*First Instructions Page View*
SecondInstructionsPageViewController	– View controller class	*Second Instructions Page View*
ThirdInstructionsPageViewController	– View controller class	*Third Instructions Page View*
FourthInstructionsPageViewController	– View controller class	*Fourth Instructions Page View*
FifthInstructionsPageViewController	– View controller class	*Fifth Instructions Page View*
SixthInstructionsPageViewController	– View controller class	*Sixth Instructions Page View*
AudioEngine	– Class that manages all the audio features, audio session, logic and components	—
GeneralLocalizer	– Class that manages the language of the application based on the localization settings	—
CurrentLanguageLocalizer	– Class that manages the current language on the scope of the application	—

Table 4.13: *MasterVoicing*'s developed code - Views

Name	Description
Main View	– Initial and primary view, that presents the contents related with the main features
Settings View	– Modal view that presents the contents of the settings' menu
Instructions View	– View that serves as a container for the other view pages of the instructions
First Instructions Page View	– View that presents the content of the first page of the instructions
Second Instructions Page View	– View that presents the content of the second page of the instructions
Third Instructions Page View	– View that presents the content of the third page of the instructions
Fourth Instructions Page View	– View that presents the content of the fourth page of the instructions
Fifth Instructions Page View	– View that presents the content of the fifth page of the instructions
Sixth Instructions Page View	– View that presents the content of the sixth page of the instructions

The multiple view controller classes used are responsible for the changes performed in their respective associated views. The *AudioEngine* class handles all the audio related manipulation and connects it to the algorithm's audio processing, through the classes in Objective-C and C++ (described previously in Section 4.3): *Algorithm-Wrapper.h*, *Algorithm-Wrapper.mm*, *Algorithm.hpp* and *Algorithm.cpp*. Adding to these files, there are also other accessory files, such as the *MasterVoicing-Bridging-Header.h*, the mandatory *Info.plist* and the application's storyboard files: *Main.storyboard* and *LaunchScreen.storyboard*.

Besides this, the application uses localization, to ensure that the UI elements change if the language changes both inside the application or on the device, which use the *GeneralLocalizer* and *CurrentLanguageLocalizer* classes and require a set of *Localizable.strings* files. The chosen language inside the application overrides the original device's language.

4.5 Evaluation

This section describes some of the tests made, after the conclusion of *Sprint 3*, to the final product of this work - the *MasterVoicing* application - and its results.

4.5.1 Audio Performance Tests

In the initial tests that were the basis of the implementation process of *MasterVoicing*, a few measures of the audio latencies were made, while the application was running, in order to compare the audio performance between Objective-C and Swift and the two different considered audio frameworks (*AudioToolbox* and *AVFoundation*), as previously stated in Section 4.2.5.

After finishing the implementation of *MasterVoicing*, the same simple audio performance tests were made to the application, in order to better understand the application's quality in terms of audio processing speed. The measured values, the audio input and output latencies[13] of the audio session, and its results are depicted in Table 4.14. The tests were made using different values for different characteristics, such as buffer duration, audio sampling frequency, number of samples of the buffer or testing with or without the algorithm processing the audio. However, the only characteristics that made an alteration in the resulting values were the sample size and sampling frequency of the audio, which is the reason they are the only variables considered in Table 4.14. It was concluded that if the audio is processed at a higher sampling frequency and corresponding buffer size, its latency values decrease slightly.

Table 4.14: Audio performance tests - Results

Sampling Frequency (Hz) | Sample Size	Input Latency	Output Latency
22050.0 | 512	0.01256 seconds	0.00794 seconds
44100.0 | 1024	0.01249 seconds	0.00630 seconds

4.5.2 Usability Testing

To complement the audio performance testing, usability tests were conducted with the purpose of evaluating the application in terms of its visual appearance and interaction. Besides the obvious goal of getting an impression of the application's usability from the possible future users of the application, another goal of these tests was to find out possible problems and/or improvements that could be made to the application.

[13] "Latency is the time it takes for a signal to travel through a system" [185].

The tests involved 15 participants[14], with an average age of 47 years, and were designed to have three different phases: *pre-test interview*, *test tasks* and *post-test questionnaire*.

Even though it is recommended that the participants of these tests have a moderate experience with the testing environment [187] [188], which, in the mobile application's case, is its operating system, it was difficult to find people with the desired characteristics. However, instead of seeing that as a drawback, it was used as an advantage, by comparing the tests results of participants with different degrees of familiarization and experience regarding iOS (7 were beginners and 8 were intermediate users).

The *pre-test interview*, whose collected data is organized in Section A.1, consisted in a set of questions to find out the characteristics of the participants and their familiarization with iOS. This phase was conducted by the facilitator of these tests (the writer of this document, with the aid of a *Facilitator Guide* document, shown in Section A.9 - written in Portuguese), in the beginning of the usability testing process. For the performance of the *test tasks*, the participant had one *Participant Guide* document (Section A.10 - also in Portuguese), which included a test orientation script, the tasks to be performed, stated in Table 4.15 and the questions which consisted in the *post-test questionnaire*. The participants were asked to classify each one of the tasks, after its completion or abandonment, in terms of difficulty, between *Easy*, *Medium* and *Hard* (to measure satisfaction, as shown in Section 4.5.2.3). The *post-test questionnaire*, that served as a way to understand the general final global perception the participant had of the application, had a rating criteria to be rated with a scale from 1 to 6, correspondent to the following meanings, in the given order: *Bad*, *Mediocre*, *Medium*, *Good*, *Very Good* and *Excellent*.

Table 4.15: Usability tests - Tasks

TN	Title	Description
1	Change pitch and volume	Alter the audio's pitch and volume.
2	Using different playback modes	Use the application in real-time mode.
3	Application's appearance	Change the color theme of the application.
4	Instructions	Find and visualize the application's instructions.
5	Recording time limitation	Change the maximum recording time to 20 seconds.

TN - Task Number

[14] According to a research made by the *Nielsen Norman Group* [186], as little as five participants are enough, in usability testing, to find almost the same usability problems that would be found with more participants.

The *post-test questionnaire* rating criteria were:

- **RC 1 -** Appearance and visual aspects;
- **RC 2 -** Ease of use, navigation and interaction;
- **RC 3 -** Features;
- **RC 4 -** Performance and speed;
- **RC 5 -** Global evaluation.

All of the tests phases were performed with the participants sitting in a quiet room, around the same table as the facilitator, who observed the participant, and interaction was established only when necessary and without compromising the credibility of the tests; each participant used a testing device, specifically an *iPhone 5S*, with the application, and were given a set of in-ear headphones for using while testing, if they thought so necessary.

The facilitator collected all the necessary data in each one of the tests, through the interview, questionnaire and observations, in order to evaluate the usability tests by using the usually considered usability metrics, which are [189]:

- **Effectiveness -** the degree to which a user can accurately accomplished goals or tasks;
- **Efficiency -** the resources spent relatively to the accurately accomplished goals or tasks;
- **Satisfaction -** the comfort and fulfillment the user feels while using the application.

The next four sections provide all the results about the evaluation of the application, extracted from all the information collected in the usability tests, which is included in Appendix A.

4.5.2.1 Effectiveness Results

The effectiveness was calculated for each task, according to the formula in Figure 4.22, whose results are shown in Figure 4.23 and show that the second and fourth tasks were successfully completed by all the participants and that the fifth task was the one that more participants abandoned.

$$Effectiveness = \frac{\text{Number of tasks completed successfully}}{\text{Total number of tasks undertaken}} \times 100\%$$

Figure 4.22: Evaluation - Effectiveness formula (Adapted from [189])

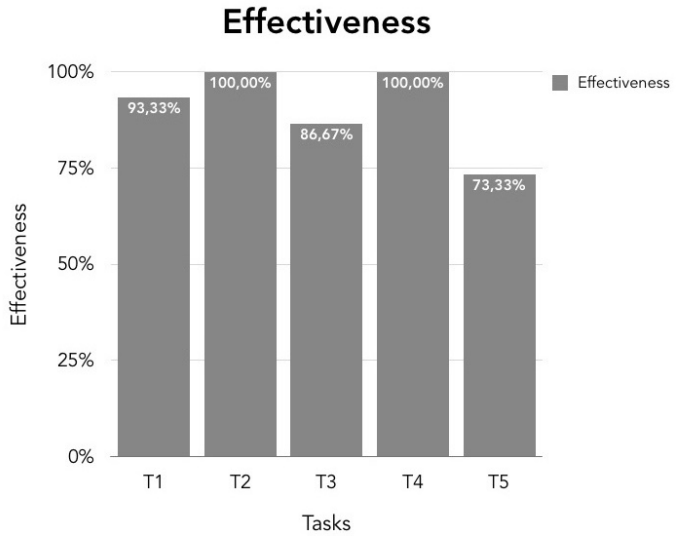


Figure 4.23: Evaluation - Effectiveness results

4.5.2.2 Efficiency Results

In terms of efficiency, two types of efficiency were calculated, according to the formulas in Figures 4.24 and 4.26[15], that correspond to the *Time Based Efficiency* and *Overall Relative Efficiency* (values depicted in Figure 4.26). The *Time Based Efficiency* has the calculated value of *0.1463 tasks/sec* for all participants, and respectively *0.0655 tasks/sec* and *0.2171 tasks/sec* for beginners and intermediate users. The *Overall Relative Efficiency* results confirm that familiarity with the platform significantly facilitates the successful completion of the tasks.

$$Time\ Based\ Efficiency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{n_{ij}}{t_{ij}}}{NR}$$

Figure 4.24: Evaluation - Time based efficiency formula (Adapted from [189])

$$Overall\ Relative\ Efficiency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N} n_{ij}t_{ij}}{\sum_{j=1}^{R}\sum_{i=1}^{N} t_{ij}} \times 100\%$$

Figure 4.25: Evaluation - Overall relative efficiency formula (Adapted from [189])

[15] In these formulas, N is the total number of tasks, R is the number of participants, n_{ij} is the result of a task i by participant j (if the participant completes the task, this value is equal to 1; if not, then is equal to 0) and t_{ij} is the time spent by participant j to complete the task or until its abandonment (the task duration) [189].

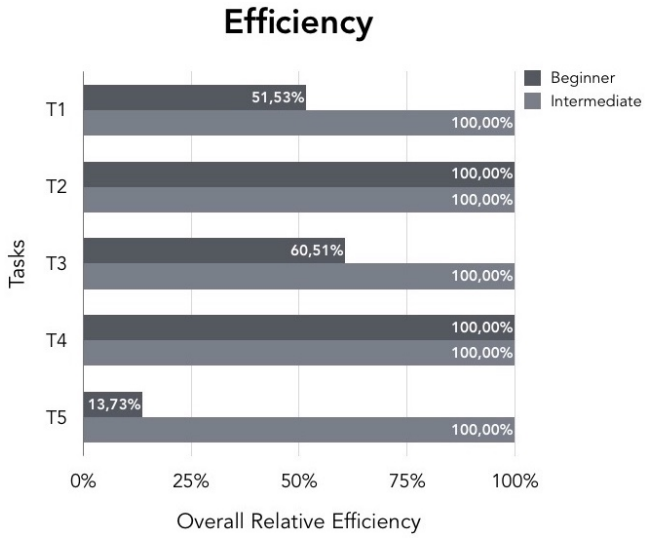


Figure 4.26: Evaluation - Overall relative efficiency results

4.5.2.3 Satisfaction Results

The *Task Level Satisfaction* was studied using the collected data from the *test tasks* (included on the tests tasks collected data, in Appendix A). These results are shown in Figure 4.27, with the opinions of the participants about each task difficulty. According to them, the first and fourth tasks were the easiest tasks and the fifth task was the hardest one to complete or understand.

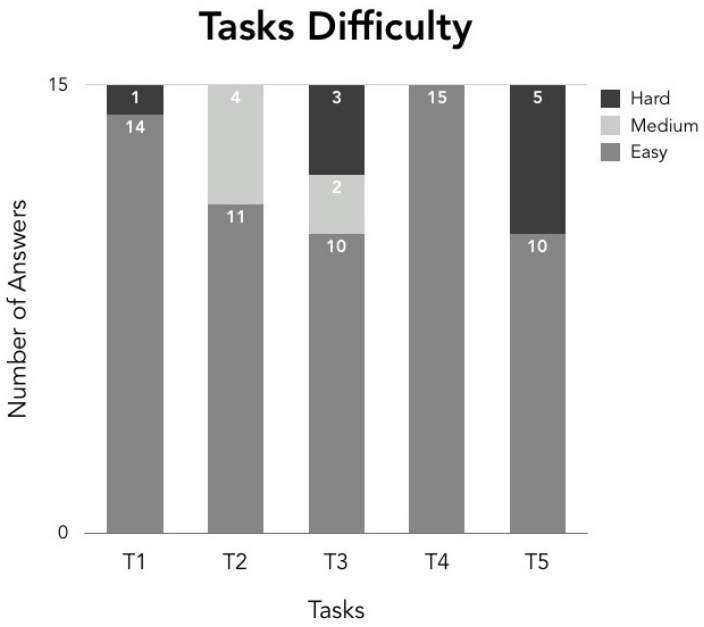


Figure 4.27: Evaluation - Task level satisfaction - Tasks difficulty

4.5.2.4 Post-Test Questionnaire Results

The mean values of the collected data in Section A.8, correspondent to the global rating of the application, are shown in Figure 4.28, along with their corresponding 95% confidence intervals (detailed in Section A.11). This results show that the participants' mean classification for the application was *Very Good* for all the rating criteria, with variations between *Good* and *Excellent*.

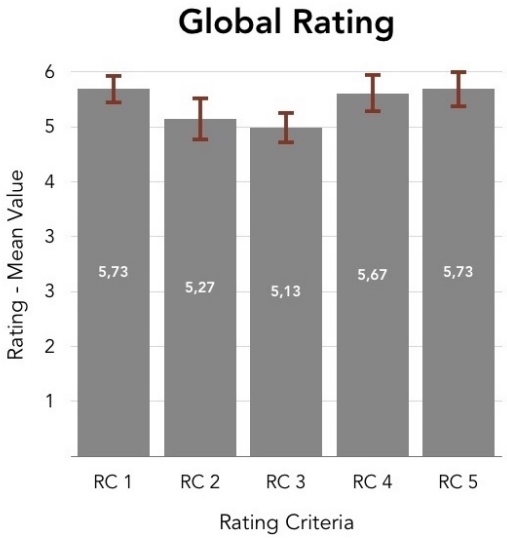


Figure 4.28: Evaluation - Post-test questionnaire - Mean values and 95% confidence intervals

The opinions and suggestions from the participants, which are described in Section A.7, unveiled some flaws and missing elements in the application, which recommend future improvements, as described in the next chapter, in Section 5.2.

4.6 Conclusions

The creation process of *MasterVoicing* was challenging, but also a great path to learn new technologies and concepts. This includes not only the novelty of learning how to develop mobile applications for iOS, but also the acquisition of knowledge about other subjects, more clearly addressed in other cycles of studies, such as audio processing. Besides that, it required the application and expansion of knowledge about previous learnt subjects, related with software engineering and human-computer interaction. Moreover, it is hoped that the creation of this application is only the beginning of a successful path for the creation of many more iOS applications to aid individuals with the particular type of speech considered in this dissertation, and that future updates will introduce new features (such as the ones suggested in 5.2) that improve its usefulness.

Chapter 5

Conclusions and Future Work

This chapter ends the work of this dissertation, which consisted in seeking the conclusion of its initial objectives, declared in Section 1.3, along with its writing, which documents the process of their completion.

Table 5.1: Dissertation objectives - Timeline

Tasks / Month	*	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep
Dissertation writing	■	■	■	■	■	■	■	■	■
Study of the problem and its solutions	■								
Further study for the state of the art	■								
iOS development knowledge acquisition		■	■	■	■	■	■	■	■
Programming languages learning		■	■	■	■	■	■	■	■
Development of a throwaway prototype				■	■	■			
Development of the application							■	■	■
Algorithm adaptation and encapsulation						■	■	■	
Algorithm verification and validation									
Evaluation of the application									■
Distribution of the application									

■ *Completed* * Dissertation Planning: course unit that prepared and started this work

5.1 Achievements

The objectives to achieve were planned monthly, between June and September of 2017, whose timeline is shown in the previous table: Table 5.1 (the achieved objectives are marked in it as *Completed*, in a green/darker color).

The timeline plan for the completion of the objectives was changed and delayed from its previous initial design, scheduled in the Dissertation Planning course unit, mainly because of two delaying situations. The first one was a delay in the initial development of the prototype, created by problems related to the complexity of the implementation of the desired audio features for the application, that revealed themselves in the initial tests and were successfully solved - this is detailed in Section 4.2. The second situation that also postponed the completion of *MasterVoicing* was a delay in the development of the algorithm, related to its complexity, which was meant to be used by the application.

As shown on Table 5.1, since the whisper-to-speech conversion algorithm intended to be used couldn't be developed on the available time constraint, as previously explained, two of the objectives were abandoned:

- Algorithm verification and validation;
- Distribution of the application.

This also means that the adaptation and encapsulation of the algorithm was made throughout June, July and August, with the creation of the prototype, in *Sprint 1* and later in *Sprint 3*, with the addition of the pitch changing slider.

Besides those two objectives, all the initial objectives were achieved.

5.2 Future Work

For future work, it is suggested the development of the abandoned features described in 4.2.7, to allow the user to share and save its recordings for later use.

From the opinions and suggestions from the participants, gathered in the usability testing, detailed in Section A.7, a set of improvements were thought for the application as possible future work:

- Accessibility improvements, such as the possibility to use a bigger font size or bigger UI elements;

- Improvements on the instructions' menu, related with content, design and navigation;
- Present the instructions' menu right after the launch of the application;
- Redesign of the settings' icon for better identification and recognition;
- Implement real-time changes in settings that produce immediate results, without requiring exiting its menu;
- Hiding the "Pitch" slider in the *Real Voice* audio processing mode;
- Improvements in alerts's animations.

Regarding other improvements and new features, it is also suggested the implementation of the following:

- Full functionality of the application using the built-in-speaker of the device (currently, real-time mode requires an external audio output, due to the difficult problem of eliminating the audio feedback resulting from having recording and playback working at the same time);
- Improvements provided by the new announced features and updates of iOS 11 for *AVFoundation*, such as better support for real-time and offline audio processing (this would allow, for example, the offline processing of an imported audio file to the application) [164].

It is hoped that the initially desired whisper-to-speech algorithm is developed in the future. In that case, the two abandoned objectives should be completed, along with the adaptation and encapsulation of the algorithm in the application, in order to give an actual real-world usefulness to the application.

5.3 Conclusions

Even thought the application does not use a voice processing algorithm related with whisper-to-speech conversion, which means it cannot be used for its intended main purpose, it consists in a complete application with several functionalities. Therefore, the application could also be used in the future with other audio processing algorithms, provided that they are adapted to the configuration of the audio buffer processing explained in Section 4.3. Even so, it has also room for improvements, regarding both its functionality and appearance.

In conclusion, it is thought that the continuation of the development of this application, including a functional whisper-to-speech algorithm, would be of great assistance for those with aphonia,

since the current solutions all have disadvantages and do not seem to be as convenient as desired by those who may benefit from them.

References

Books

[3] M. P. Robb, *Intro a guide to communication sciences and disorders*, 2nd Edition. Plural Publishing, Inc, Nov. 2013, ISBN: 9781597565424. [Online]. Available: https://www.pluralpublishing.com/publication_intro2e.htm.

[12] A. E. Aronson and D. Bless, *Clinical voice disorders*, 4th Edition. Thieme, Mar. 2009, ISBN: 9781588906625. [Online]. Available: http://www.thieme.com/books-main/speech-language-pathology/product/1796-clinical-voice-disorders.

[15] R. Rieber, *Communication disorders*, ser. Applied psycholinguistics and communication disorders. Springer US, Nov. 2013, ISBN: 9781475797602. [Online]. Available: https://books.google.pt/books?id=zcvSBwAAQBAJ.

[24] C. Jacobs, *Carcinomas of the head and neck: Evaluation and management*, ser. Cancer Treatment and Research. Springer US, Dec. 2012, ISBN: 9781461314998. [Online]. Available: https://books.google.pt/books?id=nEcIBgAAQBAJ.

[30] G. Albrecht, *Encyclopedia of disability*. SAGE Publications, Oct. 2005, ISBN: 9781452265209. [Online]. Available: https://books.google.pt/books?id=LwxzAwAAQBAJ.

[32] E. Ward and C. van As-Brooks, *Head and neck cancer: Treatment, rehabilitation, and outcomes*, 2nd Edition. Plural Publishing, Incorporated, Jul. 2014, ISBN: 9781597566599. [Online]. Available: https://books.google.pt/books?id=o9ozBwAAQBAJ.

[34] J. Freitas, A. Teixeira, M. Dias, and S. Silva, *An introduction to silent speech interfaces*, ser. SpringerBriefs in Electrical and Computer Engineering. Springer International Publishing, Aug. 2016, ISBN: 9783319401744. [Online]. Available: https://books.google.pt/books?id=mdHMDAAAQBAJ.

[130] I. Sommerville, *Software engineering*, ser. International Computer Science Series. Pearson, 2011, ISBN: 9780137053469. [Online]. Available: https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/.

[245] Mosby, *Mosby's medical dictionary*, 9th Edition, Elsevier, Ed. Elsevier Health Sciences, Jun. 2013, ISBN: 9780323112581. [Online]. Available: https://books.google.pt/books?id=aW0zkZl0JgQC.

Articles

[1] N. Roy, R. M. Merrill, S. D. Gray, and E. M. Smith, "Voice disorders in the general population: Prevalence, risk factors, and occupational impact", *The Laryngoscope*, vol. 115, no. 11, pp. 1988–1995, Nov. 2005, ISSN: 1531-4995. DOI: 10.1097/01.mlg.0000179174.32345.41. [Online]. Available: http://dx.doi.org/10.1097/01.mlg.0000179174.32345.41.

[2] N. Roy, R. M. Merrill, S. Thibeault, S. D. Gray, and E. M. Smith, "Voice disorders in teachers and the general population effects on work performance, attendance, and future career choices", *Journal of Speech, Language, and Hearing Research*, vol. 47, no. 3, pp. 542–551, Jun. 2004, ISSN: 1092-4388. DOI: 10.1044/1092-4388(2004/042). [Online]. Available: http://dx.doi.org/10.1044/1092-4388(2004/042).

[4] S. Misono, C. B. Peterson, L. Meredith, K. Banks, D. Bandyopadhyay, B. Yueh, and P. A. Frazier, "Psychosocial distress in patients presenting with voice concerns", *Journal of Voice*, vol. 28, no. 6, pp. 753–761, Nov. 2014, ISSN: 0892-1997. DOI: 10.1016/j.jvoice.2014.02.010. [Online]. Available: http://dx.doi.org/10.1016/j.jvoice.2014.02.010.

[33] Y. H. Shin and J. Seo, "Towards contactless silent speech recognition based on detection of active and visible articulators using ir-uwb radar", *Sensors*, vol. 16, no. 11, Oct. 2016, ISSN: 1424-8220. DOI: 10.3390/s16111812. [Online]. Available: http://dx.doi.org/10.3390/s16111812.

[35] B. Denby, T. Schultz, K. Honda, T. Hueber, J. Gilbert, and J. Brumberg, "Silent speech interfaces", *Speech Communication*, vol. 52, no. 4, pp. 270–287, Aug. 2009, Silent Speech Interfaces, ISSN: 0167-6393. DOI: 10.1016/j.specom.2009.08.002. [Online]. Available: http://dx.doi.org/10.1016/j.specom.2009.08.002.

[36] J. A. Gonzalez, L. A. Cheah, J. M. Gilbert, J. Bai, S. R. Ell, P. D. Green, and R. K. Moore, "A silent speech system based on permanent magnet articulography and direct synthesis", *Computer Speech & Language*, vol. 39, pp. 67–87, Feb. 2016, ISSN: 0885-2308. DOI: 10.1016/j.csl.2016.02.002. [Online]. Available: http://dx.doi.org/10.1016/j.csl.2016.02.002.

[82] I. V. Mcloughlin, H. R. Sharifzadeh, S. L. Tan, J. Li, and Y. Song, "Reconstruction of phonated speech from whispers using formant-derived plausible pitch modulation", *ACM Trans. Access. Comput.*, vol. 6, no. 4, 12:1–12:21, May 2015, ISSN: 1936-7228. DOI: 10.1145/2737724. [Online]. Available: http://doi.acm.org/10.1145/2737724.

[91] H. R. Sharifzadeh, I. V. McLoughlin, and F. Ahmadi, "Reconstruction of normal sounding speech for laryngectomy patients through a modified celp codec", *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 10, pp. 2448–2458, Oct. 2010, ISSN: 0018-9294. DOI: 10.1109/TBME.2010.2053369. [Online]. Available: http://dx.doi.org/10.1109/TBME.2010.2053369.

[137] T. Pagotto, J. A. Fabri, A. Lerario, and J. A. Gonçalves, "Scrum solo: Software process for individual development", *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–6, Jun. 2016. DOI: 10.1109/CISTI.2016.7521555. [Online]. Available: http://dx.doi.org/10.1109/CISTI.2016.7521555.

[240] H. Konno, M. Kudo, H. Imai, and M. Sugimoto, "Whisper to normal speech conversion using pitch estimated from spectrum", *Speech Communication*, vol. 83, pp. 10–20, Jul. 2016, ISSN: 0167-6393. DOI: 10.1016/j.specom.2016.07.001. [Online]. Available: http://dx.doi.org/10.1016/j.specom.2016.07.001.

[241] Z.-h. Ling, "Whisper-to-speech conversion using restricted boltzmann machine arrays", English, *Electronics Letters*, vol. 50, 1781–1782(1), 24 Nov. 2014, ISSN: 0013-5194. DOI: 10.1049/el.2014.1645. [Online]. Available: http://dx.doi.org/10.1049/el.2014.1645.

Conference Papers

[26] J. Wang, A. Samal, J. R. Green, and F. Rudzicz, "Sentence recognition from articulatory movements for silent speech interfaces", in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2012, pp. 4985–4988. DOI:

10.1109/ICASSP.2012.6289039. [Online]. Available: http://dx.doi.org/10.1109/ICASSP.2012.6289039.

[83] J. Li, I. V. McLoughlin, and Y. Song, "Reconstruction of pitch for whisper-to-speech conversion of chinese", in *Chinese Spoken Language Processing (ISCSLP), 2014 9th International Symposium on*, Sep. 2014, pp. 206–210. DOI: 10.1109/ISCSLP.2014.6936709. [Online]. Available: http://dx.doi.org/10.1109/ISCSLP.2014.6936709.

[84] Ä. T. Grozdić, B. Marković, J. Galić, and S. T. Jovičić, "Application of neural networks in whispered speech recognition", in *Telecommunications Forum (TELFOR), 2012 20th*, Nov. 2012, pp. 728–731. DOI: 10.1109/TELFOR.2012.6419311. [Online]. Available: http://dx.doi.org/10.1109/TELFOR.2012.6419311.

[85] F. Ahmadi, I. V. McLoughlin, and H. R. Sharifzadeh, "Analysis-by-synthesis method for whisper-speech reconstruction", in *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, Nov. 2008, pp. 1280–1283. DOI: 10.1109/APCCAS.2008.4746261. [Online]. Available: http://dx.doi.org/10.1109/APCCAS.2008.4746261.

Master's Thesis

[87] P. C. R. de Oliveira, "Artificial voicing of whispered speech", Master's thesis, Faculdade de Engenharia da Universidade do Porto, Jul. 2015. [Online]. Available: http://hdl.handle.net/10216/79602.

[129] RICKY DON PREUNINGER, "THE ADVANTAGES OF IMPLEMENTING SOFTWARE ENGINEERING PROCESS MODELS", Master's thesis, THE UNIVERSITY OF TEXAS AT ARLINGTON, May 2006. [Online]. Available: https://pdfs.semanticscholar.org/5c42/530b2f717ab64344d9354803ec9101079721.pdf.

[136] A. B. Hollar, "Cowboy: An Agile Programming Methodology for a Solo Programmer", Master's thesis, Virginia Commonwealth University, 2006. [Online]. Available: http://scholarscompass.vcu.edu/etd/741/.

[238] P. J. P. de Azevedo, "Vozeamento artificial de fala não vozeada", Master's thesis, Faculdade de Engenharia da Universidade do Porto, Oct. 2012. [Online]. Available: http://hdl.handle.net/10216/72539.

PhD Thesis

[90] V.-A. Tran, "Silent communication: Whispered speech-to-clear speech conversion", Theses, Institut National Polytechnique de Grenoble - INPG, 2010. [Online]. Available: https://tel.archives-ouvertes.fr/tel-00614289.

[239] R. W. Morris, "Enhancement and recognition of whispered speech", PhD thesis, School of Electrical and Computer Engineering - Georgia Institute of Technology, Atlanta, GA, USA, Aug. 2003. [Online]. Available: https://smartech.gatech.edu/bitstream/handle/1853/5425/morris_robert_w_200312_phd.pdf.

Web Pages

[6] Statista, *Number of apps available in leading app stores as of June 2016*. [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

[7] Investopedia, LLC., *Apple App Store (AAPL, GOOG)*, 2017. [Online]. Available: http://www.investopedia.com/terms/a/apple-app-store.asp.

[8] Apple Inc., *iTunes Store*, 2017. [Online]. Available: https://itunes.apple.com.

[9] About, Inc., *What Is Google Play?*, Oct. 2016. [Online]. Available: https://www.lifewire.com/what-is-google-play-1616720.

[10] Google Inc., *Google Play*, 2017. [Online]. Available: https://play.google.com/store/apps.

[11] American Speech-Language-Hearing Association (ASHA), *Definitions of communication disorders and variations [Relevant Paper]*, 1993. [Online]. Available: http://www.asha.org/policy/RP1993-00208/.

[13] EmpowHER, *Aphonia*, Apr. 2010. [Online]. Available: http://www.empowher.com/media/reference/aphonia.

[14] Advameg, Inc., *Conversion disorder*, 2017. [Online]. Available: http://www.minddisorders.com/Br-Del/Conversion-disorder.html.

[16] Dr. Elisa Shipon-Blum, *What is selective mutism?* [Online]. Available: http://www.selectivemutismcenter.org/aboutus/whatisselectivemutism.

[17] Associação Nacional das Farmácias, *Rouquidão e Afonia*, Nov. 2006. [Online]. Available: http://www.farmcentral.pt/resources/pdf/Sistema_respiratorio/rouquidaoeafonia.pdf.

[18] Thyroid Head & Neck Cancer Foundation, *Head & Neck Cancer Guide - Laryngectomy*, 2017. [Online]. Available: http://www.headandneckcancerguide.org/adults/cancer-diagnosis-treatments/surgery-and-rehabilitation/cancer-removal-surgeries/laryngectomy.

[19] NHS, *Laryngitis - Causes*, Sep. 2015. [Online]. Available: http://www.nhs.uk/Conditions/Laryngitis/Pages/Causes.aspx.

[20] Voice Foundation, *Spasmodic Dysphonia*, 2017. [Online]. Available: http://voicefoundation.org/health-science/voice-disorders/voice-disorders/spasmodic-dysphonia/.

[21] R. F. Neiman, J. R. Mountjoy, and E. L. Allen, *Myasthenia gravis focal to the larynx. Report of a case.* eng, Sep. 1975. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/1164241.

[22] eHealthMe, *Multiple sclerosis and Aphonia - from FDA reports*, 2017. [Online]. Available: http://www.ehealthme.com/cs/multiple%20sclerosis/aphonia/.

[23] eHealthMe, *Parkinson's disease and Aphonia - from FDA reports*, 2017. [Online]. Available: http://www.ehealthme.com/cs/parkinson%27s%20disease/aphonia/.

[25] Thyroid Head & Neck Cancer Foundation, *Head & Neck Cancer Guide - Speech and Swallowing Rehabilitation*, 2017. [Online]. Available: http://www.headandneckcancerguide.org/adults/cancer-diagnosis-treatments/surgery-and-rehabilitation/surgeries-to-aid-breathing-and-eating/speech-and-swallowing-rehabilitation/.

[27] American Speech-Language-Hearing Association (ASHA), *Evaluation and Treatment for Tracheoesophageal Puncture and Prosthesis: Technical Report*, 2004. [Online]. Available: http://www.asha.org/policy/TR2004-00138/.

[28] Erica Roth, *Anatomy of the Neck*, Jan. 2016. [Online]. Available: http://www.healthline.com/health/laryngectomy#Anatomy3.

[29] Memorial Sloan Kettering Cancer Center, *About Your Total Laryngectomy*, May 2016. [Online]. Available: https://www.mskcc.org/cancer-care/patient-education/about-your-total-laryngectomy.

[31] Maxi-Aids, *Trutone Electrolarynx*, 2017. [Online]. Available: https://www.amazon.com/Maxi-Aids-Trutone-Electrolarynx/dp/B003E6LNTE.

[37] Acapela Group, *my-own-voice*. [Online]. Available: http://www.acapela-group.com/voices/voice-replacement/.

[51] AssistiveWare, *PolyPredix*, 2017. [Online]. Available: http://www.assistiveware.com/innovation/polypredix.

[54] Tobii Dynavox, *M-8*, 2017. [Online]. Available: https://www.tobiidynavox.com/devices/Multi-Access-Devices/M-8/.

[55] ABILIA, *Lightwriter SL40*, 2011. [Online]. Available: http://www.toby-churchill.com/products/lightwriter-sl40/.

[56] American Speech-Language-Hearing Association (ASHA), *Augmentative and Alternative Communication (AAC)*, 2017. [Online]. Available: http://www.asha.org/public/speech/disorders/AAC/.

[57] Tobii Dynavox, *T15*, 2017. [Online]. Available: https://www.tobiidynavox.com/devices/Multi-Access-Devices/T15-1/.

[58] Tobii Dynavox, *DynaVoxDynaWrite 2.0*, 2015. [Online]. Available: http://www.dynavoxtech.com/products/dynawrite/.

[59] Tobii Dynavox, *I-15+*, 2017. [Online]. Available: https://www.tobiidynavox.com/devices/Eye-Gaze-Devices/I-15-with-Communicator/.

[60] UNESCO, *Statistics on Literacy*, 2016. [Online]. Available: http://www.unesco.org/new/en/education/themes/education-building-blocks/literacy/resources/statistics.

[81] Larry Medwetsky, *Mobile Device Apps for People with Hearing Loss*, Sep. 2015. [Online]. Available: http://www.hearingloss.org/sites/default/files/docs/HLM_SeptOct2015_Medwetsky.pdf.

[88] Dictionary.com, LLC., *vocoder*, 2017. [Online]. Available: http://www.dictionary.com/browse/vocoder.

[89] Dictionary.com, LLC., *phoneme*, 2017. [Online]. Available: http://www.dictionary.com/browse/phoneme.

[94] Yoni Heisler, *The history and evolution of iOS, from the original iPhone to iOS 9*, Feb. 2016. [Online]. Available: http://bgr.com/2016/02/12/ios-history-iphone-features-evolution/.

[96] Rhiannon Williams, *Apple iOS: a brief history*, Sep. 2015. [Online]. Available: http://www.telegraph.co.uk/technology/apple/11068420/Apple-iOS-a-brief-history.html.

[97] 9to5 Staff, *Jobs' original vision for the iPhone: No third-party native apps*, Oct. 2011. [Online]. Available: https://9to5mac.com/2011/10/21/jobs-original-vision-for-the-iphone-no-third-party-native-apps/.

[98] Statista, *Number of apps available in leading app stores as of March 2017*, 2017. [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

[115] JUDSON BANDY, *Objective-C and Swift: Their History and Where iOS Development is Going*, Feb. 2017. [Online]. Available: http://hangzone.com/objective-c-swift-history-apple-development-going/.

[119] Apple Inc., *About Swift*, 2017. [Online]. Available: https://swift.org/about/.

[121] Open Source Initiative, *The Open Source Definition (Annotated)*, 2017. [Online]. Available: https://opensource.org/osd-annotated.

[128] Atul Shukla, *Top 6 Challenges in Software Development*, Apr. 2016. [Online]. Available: https://www.gate6.com/blog/top-6-challenges-software-development.

[131] VersionOne, Inc., *11th Annual State of Agile Report*, Apr. 2017. [Online]. Available: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2.

[132] Nielsen Norman Group - KARA PERNICE, *UX Prototypes: Low Fidelity vs. High Fidelity*, Dec. 2016. [Online]. Available: https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/.

[133] tutorialspoint, *SDLC - Software Prototype Model*, 2017. [Online]. Available: https://www.tutorialspoint.com/sdlc/sdlc_software_prototyping.htm.

[134] James C. Helm, Ph.D., P.E., *Methods for Software Prototyping*. [Online]. Available: http://sce.uhcl.edu/helm/REQ_ENG_WEB/My-Files/mod4/Software_Prototyping.pdf.

[135] Ken Schwaber and Jeff Sutherland, *The Scrum Guide*, Jul. 2016. [Online]. Available: http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100.

[138] Atlassian, *Epics, stories, versions, and sprints*, 2017. [Online]. Available: https://www.atlassian.com/agile/delivery-vehicles.

[139] Jonathan Rasmusson, *Burndown Charts*. [Online]. Available: http://www.agilenutshell.com/burndown.

[141] Google Inc., *Google Search*, 2017. [Online]. Available: https://www.google.com.

[142] Stack Exchange Inc., *Developer Survey Results 2017*, 2017. [Online]. Available: https://insights.stackoverflow.com/survey/2017.

[143] Razeware LLC., *Tutorial Archive*, 2016. [Online]. Available: https://www.raywenderlich.com/tutorial-archive.

[144] Code School LLC., *Try Objective-C*, 2017. [Online]. Available: https://www.codeschool.com/courses/try-objective-c.

[145] TechTarget, *iTunes U*, 2017. [Online]. Available: http://whatis.techtarget.com/definition/iTunes-U.

[148] Udemy, Inc., *Search results for "ios"*, 2017. [Online]. Available: https://www.udemy.com/courses/search/?q=ios.

[149] Lynda.com, Inc., *iOS Training and Tutorials*, 2017. [Online]. Available: https://www.lynda.com/iOS-training-tutorials/413-0.html.

[150] Razeware LLC., *Tutorials - iOS*, 2016. [Online]. Available: https://www.raywenderlich.com/category/ios.

[151] Razeware LLC., *Tutorials - Swift*, 2016. [Online]. Available: https://www.raywenderlich.com/category/swift.

[152] Code School LLC., *Paths - iOS*, 2017. [Online]. Available: https://www.codeschool.com/learn/ios.

[153] Coursera Inc., *Launch Your Career in iOS*, 2017. [Online]. Available: https://pt.coursera.org/specializations/app-development.

[154] We Heart Swift, *We Heart Swift*, 2017. [Online]. Available: https://www.weheartswift.com.

[155] Magnus, *little bites of cocoa*, 2015. [Online]. Available: https://littlebitesofcocoa.com.

[156] Zaheer, *LearnSwift.tips*, 2017. [Online]. Available: http://www.learnswift.tips/.

[157] @bardonadam, *iOS Cookies*, 2017. [Online]. Available: http://www.ioscookies.com.

[158] GitHub Guides, *Hello World*, Apr. 2016. [Online]. Available: https://guides.github.com/activities/hello-world/#what.

[159] GitHub, Inc., *Search language:Swift*, 2017. [Online]. Available: https://github.com/search?l=&p=1&q=language%3ASwift&ref=advsearch&type=Repositories&utf8=%C3%A2%C2%9C%C2%93.

[160] Udemy, Inc., *Udemy - About*, 2017. [Online]. Available: https://about.udemy.com.

[161] Udemy, Inc., *Udemy*, 2017. [Online]. Available: https://www.udemy.com.

[162] Udemy, Inc., *Udemy - iOS 10 and Xcode 8 - Complete Swift 3 & Objective-C Course*, 2017. [Online]. Available: https://www.udemy.com/ios-10-xcode-8/.

[163] Udemy, Inc., *Udemy - The Complete iOS 10 & Swift 3 Developer Course*, 2017. [Online]. Available: https://www.udemy.com/complete-ios-10-developer-course/.

[166] GitHub, Inc., *ooper-shlab / aurioTouch2.0-Swift*, 2017. [Online]. Available: https://github.com/ooper-shlab/aurioTouch2.0-Swift.

[170] GitHub, Inc., *ooper-shlab / AVAEMixerSample-Swift*, 2017. [Online]. Available: https://github.com/ooper-shlab/AVAEMixerSample-Swift.

[172] GitHub, Inc., *SwiftArchitect / SO-32541268*, 2017. [Online]. Available: https://github.com/SwiftArchitect/SO-32541268.

[173] Aurelius Prochazka, *AUDIOKIT*, 2017. [Online]. Available: http://audiokit.io.

[174] Zynaptiq GmbH, *Stephan Bernsee's Blog - Download*, 2017. [Online]. Available: http://blogs.zynaptiq.com/bernsee/download/.

[175] Apple Inc., *Keynote*, 2017. [Online]. Available: https://www.apple.com/keynote/.

[176] ZenHub, *ZenHub*, 2017. [Online]. Available: https://www.zenhub.com.

[178] Figma, *Figma - Turn Ideas into Products Faster*. [Online]. Available: https://www.figma.com/.

[179] Great Simple Studio., *iOS 10 GUI - iPhone & iPad*, May 2017. [Online]. Available: http://iosdesignkit.io/ios-gui/.

[180] Medium - ZenHub, *How to use GitHub for agile project management*, May 2016. [Online]. Available: https://medium.com/@ZenHub/how-to-use-github-for-agile-project-management-2c5b0ec1dad6.

[181] Zynaptiq GmbH, *Pitch Shifting Using The Fourier Transform*, Sep. 1999. [Online]. Available: http://blogs.zynaptiq.com/bernsee/pitch-shifting-using-the-ft/.

[182] Nielsen Norman Group - AURORA HARLEY, *Instructional Overlays and Coach Marks for Mobile Apps*, Feb. 2014. [Online]. Available: https://www.nngroup.com/articles/mobile-instructional-overlay/.

[183] Coolors, *Coolors- Generator*. [Online]. Available: https://coolors.co/app.

[184] Pixelmator Team, *Pixelmator for Mac*. [Online]. Available: http://www.pixelmator.com/mac/.

[185] NDK, *Audio Latency*. [Online]. Available: https://developer.android.com/ndk/guides/audio/audio-latency.html.

[186] Nielsen Norman Group - JAKOB NIELSEN, *How Many Test Users in a Usability Study?*, Jun. 2012. [Online]. Available: https://www.nngroup.com/articles/how-many-test-users/.

[187] UsabilityGeek - Justin Mifsud, *Usability Testing Of Mobile Applications: A Step-By-Step Guide*, Mar. 2016. [Online]. Available: http://usabilitygeek.com/usability-testing-mobile-applications/.

[188] Nielsen Norman Group - RALUCA BUDIU, *Usability Testing for Mobile Is Easy*, Feb. 2014. [Online]. Available: https://www.nngroup.com/articles/mobile-usability-testing/.

[189] UsabilityGeek - Justin Mifsud, *Usability Metrics – A Guide To Quantify The Usability Of Any System*, Jun. 2015. [Online]. Available: http://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/.

[192] ShareLaTeX, *Code listing*, 2017. [Online]. Available: https://www.sharelatex.com/learn/Code_listing#!#Introduction.

[193] MakeAppIcon powered by Skygear, *Best icon resizer for mobile app developers.* [Online]. Available: https://makeappicon.com.

[196] Wayne Media LLC, *NSUserDefaults — A Swift Introduction*, Dec. 2014. [Online]. Available: http://www.codingexplorer.com/nsuserdefaults-a-swift-introduction/.

[197] K Harrison, *Using a Launch Screen Storyboard*, Dec. 2014. [Online]. Available: https://useyourloaf.com/blog/using-a-launch-screen-storyboard/.

[198] Atomic Object LLC - JEFF BURT, *How to Use UIPageViewController in Swift*, Dec. 2015. [Online]. Available: https://spin.atomicobject.com/2015/12/23/swift-uipageviewcontroller-tutorial/.

[199] Atomic Object LLC - JEFF BURT, *How to Move Page Dots in a UIPageViewController*, Feb. 2016. [Online]. Available: https://spin.atomicobject.com/2016/02/11/move-uipageviewcontroller-dots/.

[200] ONESKY INC., *INTRODUCTION TO IOS LOCALIZATION*, 2015. [Online]. Available: https://www.oneskyapp.com/academy/learn-ios-localization/.

[201] GitHub, Inc., *codepath / ios_guides - UserDefaults*, 2017. [Online]. Available: https://github.com/codepath/ios_guides/wiki/UserDefaults.

[202] GitHub, Inc., *MoathOthman / Localization102*, 2017. [Online]. Available: https://github.com/MoathOthman/Localization102.

[203] GitHub, Inc., *jeffaburt / UIPageViewController-Post*, 2017. [Online]. Available: https://github.com/jeffaburt/UIPageViewController-Post.

[206] Nielsen Norman Group - AMY SCHADE, *Write Better Qualitative Usability Tasks: Top 10 Mistakes to Avoid*, Apr. 2017. [Online]. Available: https://www.nngroup.com/articles/better-usability-tasks/.

[207] Nielsen Norman Group - AURORA HARLEY, *Usability Testing of Icons*, Feb. 2016. [Online]. Available: https://www.nngroup.com/articles/icon-testing/.

[208] YouTube - Mark Moeykens, *Customizing UIPickerView (iOS, Xcode 8, Swift 3)*, Apr. 2017. [Online]. Available: https://www.youtube.com/watch?v=6Qd3CdWYeJ8.

[209] GitHub, Inc., *awseeley / Swift-Square-to-Circle-Transformation-Tutorial*, 2017. [Online]. Available: https://github.com/awseeley/Swift-Square-to-Circle-Transformation-Tutorial/blob/master/SquareToCircle/ViewController.swift.

[210] Medium - moath othman, *Working with Internationalization and Localization in swift*, Mar. 2016. [Online]. Available: https://medium.com/if-let-swift-programming/working-with-localization-in-swift-4a87f0d393a4.

[211] Andrew Bancroft, *Fade In / Out Animations as Class Extensions in Swift*, Jul. 2014. [Online]. Available: https://www.andrewcbancroft.com/2014/07/27/fade-in-out-animations-as-class-extensions-with-swift/.

[212] ObjectCoder, *Detecting if Headphones are Plugged-in to an iOS Device in Swift*, Apr. 2015. [Online]. Available: https://objectcoder.com/2015/04/09/detecting-if-headphones-are-plugged-in-to-an-ios-device-in-swift/.

[213] MAKING APP PIE - Steven Lipton, *Swift Swift: Formatting a UIPickerView*, Oct. 2014. [Online]. Available: https://makeapppie.com/2014/10/21/swift-swift-formatting-a-uipickerview/.

[214] IOSCREATOR - Arthur Knopper, *Detect Shake Gestures iOS Tutorial*, Apr. 2017. [Online]. Available: https://www.ioscreator.com/tutorials/detect-shake-gestures-ios-tutorial-ios10.

[215] Hacking with Swift - Paul Hudson, *How to use light text color in the status bar*, 2017. [Online]. Available: https://www.hackingwithswift.com/example-code/uikit/how-to-use-light-text-color-in-the-status-bar.

[216] Medium - Ben Tranter, *Using GitHub with Xcode 6*, Feb. 2015. [Online]. Available: https://medium.com/@0xben/using-github-with-xcode-6-8208b92c7a60.

[217] Matt Butler and Paige Paquette, *Better Software & Stronger Teams - Project Management for GitHub*, 2016. [Online]. Available: https://www.zenhub.com/github-project-management.pdf?source=onboarding.

[232] Tables Generator, *LaTeX Table Generator*. [Online]. Available: http://www.tablesgenerator.com.

[242] WebWhispers, *Talking Again*, 2017. [Online]. Available: http://www.webwhispers.org/library/talking-again.asp.

[243] My Voice, *Methods of speaking after laryngectomy*. [Online]. Available: http://dribrook.blogspot.pt/p/basic-skills-for-new-laryngectomees.html.

[244] WebWhispers, *Text to Speech Apps*, 2017. [Online]. Available: http://www.webwhispers.org/library/TexttoSpeechApps.asp.

[246] Mary Cresse, *Loss of Voice*, 2017. [Online]. Available: http://www.svmh.com/health/content.aspx?chunkiid=432303.

App Store Applications

[5] SEEGNAL, *MasterPitch*. [Online]. Available: http://www.masterpitchapp.com/Home/MasterPitch.html.

[38] Intuary, Inc., *Verbally Premium*, 2017. [Online]. Available: http://www.verballyapp.com/premium.html.

[39] Loic Verrall, *Speak - Text To Speech*, Jan. 2017. [Online]. Available: https://itunes.apple.com/us/app/speak-text-to-speech/id806819422?mt=8.

[40] Gwyn Durbridge, *Text to Speech - Voice Synthesiser*, Dec. 2016. [Online]. Available: https://itunes.apple.com/us/app/text-to-speech-voice-synthesiser/id712104788?mt=8.

[42] Chun Kai Lau, *Aloud! Texto em fala (TTS)*, Sep. 2016. [Online]. Available: https://itunes.apple.com/pt/app/aloud!-texto-em-fala-tts/id852033350?mt=8.

[47] AssistiveWare, *Proloquo2Go*, 2017. [Online]. Available: http://www.assistiveware.com/product/proloquo2go.

[48] Darrin Altman, *Talk For Me - Text to Speech*, Jan. 2017. [Online]. Available: https://itunes.apple.com/us/app/talk-for-me-text-to-speech/id975096888?mt=8.

[49] Touch Voice, *Touch Voice - The Medical Speaking App*. [Online]. Available: https://touch-voice.com/marketing/.

[50] Intuary, Inc., *Features of Verbally*, 2017. [Online]. Available: http://www.verballyapp.com/features.html.

[53] NaturalSoft Limited, *NaturalReader Mobile Text-to-Speech Reader*, 2017. [Online]. Available: https://www.naturalreaders.com/app/.

[61] Microsoft Corporation, *Microsoft Translator*, 2017. [Online]. Available: https://www.microsoft.com/en-us/translator/default.aspx.

[62] Apalon Apps, *Fale e Traduza - Tradutor de Texto e voz Gratuito*, Feb. 2017. [Online]. Available: https://itunes.apple.com/pt/app/fale-e-traduza-tradutor-texto/id804641004?mt=8.

[66] mikrosonic, *RoboVox - Voice Changer*, 2014. [Online]. Available: http://www.mikrosonic.com/robovox.

[68] Lylavie, LLC, *Voicy Phone - Prank Dial Call with Voice Changer*, Sep. 2016. [Online]. Available: https://itunes.apple.com/pt/app/voicy-phone-prank-dial-call/id898889879?mt=8.

[71] Voicemod, *Funny Call - Modificador de voz para ligações engraçadas by Voicemod*, Jul. 2016. [Online]. Available: https://itunes.apple.com/pt/app/funny-call-modificador-voz/id392640258?mt=8.

[72] Goran Jankovic, *VoiceFun - Livre Cambiador De Voz & Som Editor App Para Transformar*, Jun. 2016. [Online]. Available: https://itunes.apple.com/pt/app/voicefun-livre-cambiador-voz/id1128099623?mt=8.

[75] PSOFT, *EffecTalk*, Oct. 2014. [Online]. Available: https://itunes.apple.com/us/app/effectalk/id556787220?l=us%5C&ls=1%5C&mt=8.

[77] IT ForYou, *Petralex - Hear It Clear*, 2017. [Online]. Available: http://petralex.pro.

[78] Bxtel LLC, *Enhanced Ears Hearing Aid*, Nov. 2015. [Online]. Available: https://itunes.apple.com/pt/app/enhanced-ears-hearing-aid/id950449570?mt=8.

[79] ExSilent B.V., *HearYouNow – Your personal sound amplifier*, Oct. 2012. [Online]. Available: https://itunes.apple.com/pt/app/hearyounow-your-personal-sound/id569522474?mt=8.

[80] Overpass Ltd., *Ear Spy - The Ultimate Eavesdropping App*, 2017. [Online]. Available: http://www.overpass.co.uk/app/ear-spy/.

Google Play Applications

[41] Code Factory, *Voz Vocalizer TTS (Português)*, Oct. 2015. [Online]. Available: https://play.google.com/store/apps/details?id=es.codefactory.vocalizertts.

[43] TK Solution, *Text to Speech (TTS)*, Oct. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.textsprecher&hl=pt_PT.

[44] Simply Complex Apps, *Tell Me - Text To Speech*, Jul. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.simplycomplexapps.ASTellme.

[45] HE SOFT, *T2S:Leia em voz alta o texto*, Oct. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=hesoft.T2S.

[46] Abast Multimèdia, *Talk - Text to Voice FREE*, Oct. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.ktix007.talk&hl=pt%5C_PT.

[49] Touch Voice, *Touch Voice - The Medical Speaking App*. [Online]. Available: https://touch-voice.com/marketing/.

[52] noinnion, *Voice Reading (Read aloud)*, Aug. 2013. [Online]. Available: https://play.google.com/store/apps/details?id=com.noinnion.android.voicereading.

[53] NaturalSoft Limited, *NaturalReader Mobile Text-to-Speech Reader*, 2017. [Online]. Available: https://www.naturalreaders.com/app/.

[61] Microsoft Corporation, *Microsoft Translator*, 2017. [Online]. Available: https://www.microsoft.com/en-us/translator/default.aspx.

[63] Google Inc., *Google Tradutor*, Jan. 2017. [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.translate.

[64] Androbaby, *Mudar a voz*, Sep. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.androbaby.voicechanger.

[65] Flagman-Soft, *Real time voice changer Lite*, Aug. 2015. [Online]. Available: https://play.google.com/store/apps/details?id=com.flagmansoft.voicefunlite.

[66] mikrosonic, *RoboVox - Voice Changer*, 2014. [Online]. Available: http://www.mikrosonic.com/robovox.

[67] Baviux, *Muda voz para crianças*, Feb. 2017. [Online]. Available: https://play.google.com/store/apps/details?id=com.baviux.voicechanger.kids.

[69] BARapps, *FunCall voice changer in call*, Dec. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.rami_bar.fun_call.

[70] Zenitalk VOIP, call with us, *Voice Changer chamando*, Mar. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.weirdvoice.

[73] e3games, *Mudar a Voz*, Sep. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.e3games.voicechanger.

[74] Juergen Hock, *Voicesmith*, Feb. 2014. [Online]. Available: https://play.google.com/store/apps/details?id=de.jurihock.voicesmith.

[76] Lemberg Solutions, *Hearing Aid with Replay (Lite)*, Jan. 2014. [Online]. Available: https://play.google.com/store/apps/details?id=com.ls.soundamplifier.

[77] IT ForYou, *Petralex - Hear It Clear*, 2017. [Online]. Available: http://petralex.pro.

[80] Overpass Ltd., *Ear Spy - The Ultimate Eavesdropping App*, 2017. [Online]. Available: http://www.overpass.co.uk/app/ear-spy/.

Apple Developer Website

[92] Apple Inc., *iOS Technology Overview - Introduction*, 2014. [Online]. Available: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html.

[93] Apple Inc., *Apple Developer Homepage*, 2017. [Online]. Available: https://developer.apple.com.

[95] Apple Inc., *What's New in Xcode 9*, 2017. [Online]. Available: https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/WhatsNewXcode/xcode_9/xcode_9.html#//apple_ref/doc/uid/TP40004626-CH8-SW1.

[99] Apple Inc., *What's New in iOS 11*, 2017. [Online]. Available: https://developer.apple.com/ios/.

[100] Apple Inc., *iOS Technology Overview - Cocoa Touch Layer*, 2014. [Online]. Available: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html#//apple_ref/doc/uid/TP40007898-CH3-SW1.

[101] Apple Inc., *iOS Technology Overview - Media Layer*, 2014. [Online]. Available: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/MediaLayer/MediaLayer.html#//apple_ref/doc/uid/TP40007898-CH9-SW4.

[102] Apple Inc., *iOS Technology Overview - Core Services Layer*, 2014. [Online]. Available: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreServicesLayer/CoreServicesLayer.html#//apple_ref/doc/uid/TP40007898-CH10-SW5.

[103] Apple Inc., *iOS Technology Overview - Core OS Layer*, 2014. [Online]. Available: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreOSLayer/CoreOSLayer.html#//apple_ref/doc/uid/TP40007898-CH11-SW1.

[104] Apple Inc., *SDK Compatibility Guide - Introduction*, 2010. [Online]. Available: https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/cross_development/Introduction/Introduction.html.

[105] Apple Inc., *App Programming Guide for iOS - Expected App Behaviors*, 2017. [Online]. Available: https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/ExpectedAppBehaviors/

ExpectedAppBehaviors.html#//apple_ref/doc/uid/TP40007072-CH3-SW2.

[106] Apple Inc., *Bundle Programming Guide - Application Bundles*, 2017. [Online]. Available: https://developer.apple.com/library/content/documentation/CoreFoundation/Conceptual/CFBundles/BundleTypes/BundleTypes.html#//apple_ref/doc/uid/10000123i-CH101-SW13.

[107] Apple Inc., *Start Developing iOS Apps (Swift) - Glossary*, 2017. [Online]. Available: https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/GlossaryDefinitions.html#//apple_ref/doc/uid/TP40015214-CH12-SW1.

[108] Apple Inc., *Debugging with Xcode - About Debugging with Xcode*, 2017. [Online]. Available: https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/debugging_with_xcode/chapters/about_debugging_w_xcode.html#//apple_ref/doc/uid/TP40015022.

[109] Apple Inc., *Apple Developer Documentation*, 2017. [Online]. Available: https://developer.apple.com/documentation/.

[110] Apple Inc., *Simulator User Guide - Introduction*, 2016. [Online]. Available: https://developer.apple.com/library/content/documentation/IDEs/Conceptual/iOS_Simulator_Guide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40012848-CH1-SW1.

[111] Apple Inc., *Instruments User Guide - About Instruments*, 2017. [Online]. Available: https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/index.html#//apple_ref/doc/uid/TP40004652.

[112] Apple Inc., *Xcode - Interface Builder Built-In*, 2017. [Online]. Available: https://developer.apple.com/xcode/interface-builder/.

[113] Apple Inc., *Debugging with Xcode - Debugging Tools*, 2017. [Online]. Available: https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/debugging_with_xcode/chapters/debugging_tools.html#//apple_ref/doc/uid/TP40015022-CH8-SW1.

[114] Apple Inc., *Testing with Xcode - Quick Start*, 2017. [Online]. Available: https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/testing_with_xcode/chapters/02-quick_start.html#//apple_ref/doc/uid/TP40014132-CH2-SW1.

[116] Apple Inc., *Using Swift with Cocoa and Objective-Ct*, 2017. [Online]. Available: https://developer.apple.com/library/content/documentation/Swift/Conceptual/BuildingCocoaApps/index.html#//apple_ref/doc/uid/TP40014216.

[117] Apple Inc., *App Store*, 2017. [Online]. Available: https://developer.apple.com/support/app-store/.

[118] Apple Inc., *Swift 4*, 2017. [Online]. Available: https://developer.apple.com/swift/.

[120] Apple Inc., *The Swift Programming Language - About Swift*, 2017. [Online]. Available: https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/index.html#//apple_ref/doc/uid/TP40014097.

[122] Apple Inc., *Swift Playgrounds*, 2017. [Online]. Available: https://www.apple.com/swift/playgrounds/.

[123] Apple Inc., *Apple Developer Program - Program Membership Details*, 2017. [Online]. Available: https://developer.apple.com/programs/whats-included/.

[124] Apple Inc., *App Distribution Quick Start - Adding Your Account to Xcode*, 2016. [Online]. Available: https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppStoreDistributionTutorial/AddingYourAccounttoXcode/AddingYourAccounttoXcode.html#//apple_ref/doc/uid/TP40013839-CH40-SW1.

[125] Apple Inc., *iTunes Connect Developer Guide - Introduction*, 2017. [Online]. Available: https://developer.apple.com/library/content/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/Chapters/OverviewofiTunesConn
html#//apple_ref/doc/uid/TP40011225-CH12-SW1.

[126] Apple Inc., *App Review*, 2017. [Online]. Available: https://developer.apple.com/app-store/review/.

[127] Apple Inc., *Everyone Can Code*, 2017. [Online]. Available: https://www.apple.com/everyone-can-code/.

[140] Apple Inc., *Guides and Sample Code*, 2017. [Online]. Available: https://developer.apple.com/library/content/navigation/.

[146] Apple Inc., *iTunes U*, 2017. [Online]. Available: https://www.apple.com/education/itunes-u/.

[147] Apple Inc., *Swift - Resources*, 2017. [Online]. Available: https://developer.apple.com/swift/resources/.

[164] Apple Inc., *What's New in Audio - Presentation Slides (PDF)*, 2017. [Online]. Available: https://devstreaming-cdn.apple.com/videos/wwdc/2017/501fo36iwi2moz2l222/501/501_whats_new_in_audio.pdf.

[165] Apple Inc., *aurioTouch*, 2016. [Online]. Available: https://developer.apple.com/library/content/samplecode/aurioTouch/Introduction/Intro.html.

[167] Apple Inc., *AudioUnit*, 2017. [Online]. Available: https://developer.apple.com/documentation/audiounit.

[168] Apple Inc., *Core Audio Overview*, Feb. 2014. [Online]. Available: https://developer.apple.com/library/content/documentation/MusicAudio/Conceptual/CoreAudioOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40003577-CH1-SW1.

[169] Apple Inc., *Using AVAudioEngine for Playback, Mixing and Recording (AVAEMixerSample)*, 2017. [Online]. Available: https://developer.apple.com/library/content/samplecode/AVAEMixerSample/Introduction/Intro.html.

[171] Apple Inc., *AVAudioEngine*, 2017. [Online]. Available: https://developer.apple.com/documentation/avfoundation/avaudioengine.

[177] Apple Inc., *Human Interface Guidelines*, 2017. [Online]. Available: https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/.

Stack Exchange Website

[190] Stack Exchange Inc., *Changing navigation bar color in Swift*, 2017. [Online]. Available: https://stackoverflow.com/questions/24687238/changing-navigation-bar-color-in-swift#24687554.

[191] Stack Exchange Inc., *Chapter name in the header with \chapter**, 2017. [Online]. Available: https://tex.stackexchange.com/questions/89914/chapter-name-in-the-header-with-chapter.

[194] Stack Exchange Inc., *How to set a launch screen image in Xcode*, 2017. [Online]. Available: https://stackoverflow.com/questions/31025261/how-to-set-a-launch-screen-image-in-xcode.

[195] Stack Exchange Inc., *Change iOS app's language on the fly*, 2017. [Online]. Available: https://stackoverflow.com/questions/6150576/change-ios-apps-language-on-the-fly.

[204] Stack Exchange Inc., *How to change the background color of a UIButton while it's highlighted?*, 2017. [Online]. Available: https://stackoverflow.com/questions/14523348/how-to-change-the-background-color-of-a-uibutton-while-its-highlighted.

[205] Stack Exchange Inc., *Swift UIAlertController -> ActionSheet iPad iOS8 Crashes*, 2017. [Online]. Available: https://stackoverflow.com/questions/26039229/swift-uialertcontroller-actionsheet-ipad-ios8-crashes#26040876.

[218] Stack Exchange Inc., *How to display song currenttime and duration by using AVAudioPlayerNode*, 2017. [Online]. Available: https://stackoverflow.com/questions/26574459/how-to-display-song-currenttime-and-duration-by-using-avaudioplayernode.

[219] Stack Exchange Inc., *How to create dispatch queue in Swift 3*, 2017. [Online]. Available: https://stackoverflow.com/questions/37805885/how-to-create-dispatch-queue-in-swift-3#37806522.

[220] Stack Exchange Inc., *How to use background thread in swift?*, 2017. [Online]. Available: https://stackoverflow.com/questions/24056205/how-to-use-background-thread-in-swift#25070476.

[221] Stack Exchange Inc., *Tap Mic Input Using AVAudioEngine in Swift*, 2017. [Online]. Available: https://stackoverflow.com/questions/27203622/tap-mic-input-using-avaudioengine-in-swift.

[222] Stack Exchange Inc., *CATransform3D Key Paths with #keyPath*, 2017. [Online]. Available: https://stackoverflow.com/questions/41102217/catransform3d-key-paths-with-keypath.

[223] Stack Exchange Inc., *CABasicAnimation chaining not working*, 2017. [Online]. Available: https://stackoverflow.com/questions/39076912/cabasicanimation-chaining-not-working#39077557.

[224] Stack Exchange Inc., *How can I create an CABasicAnimation for multiple properties?*, 2017. [Online]. Available: https://stackoverflow.com/questions/10938223/how-can-i-create-an-cabasicanimation-for-multiple-properties.

[225] Stack Exchange Inc., *How to Apply Gradient to background view of iOS Swift App*, 2017. [Online]. Available: https://stackoverflow.com/questions/24380535/how-to-apply-gradient-to-background-view-of-ios-swift-app#24380968.

[226] Stack Exchange Inc., *How to change line color in tabular?*, 2017. [Online]. Available: https://tex.stackexchange.com/questions/40666/how-to-change-line-color-in-tabular#40667.

[227] Stack Exchange Inc., *How to connect the audio to bluetooth when playing using AVPlayer*, 2017. [Online]. Available: https://stackoverflow.com/questions/31133636/how-to-connect-the-audio-to-bluetooth-when-playing-using-avplayer#31242669.

[228] Stack Exchange Inc., *Swift 2, warning: could not load any Objective-C class information from the dyld shared cache*, 2017. [Online]. Available: https://stackoverflow.com/questions/31431287/swift-2-warning-could-not-load-any-objective-c-class-information-from-the-dyld.

[229] Stack Exchange Inc., *How to install a tap on the default inputNode of AudioEngine and write to file*, 2017. [Online]. Available: https://stackoverflow.com/questions/29762598/how-to-install-a-tap-on-the-default-inputnode-of-audioengine-and-write-to-file.

[230] Stack Exchange Inc., *How to change the color of UIPickerView Selector*, 2017. [Online]. Available: https://stackoverflow.com/questions/1466346/how-to-change-the-color-of-uipickerview-selector#26851981.

[231] Stack Exchange Inc., *How to change UINavigationItem font?*, 2017. [Online]. Available: https://stackoverflow.com/questions/28795735/how-to-change-uinavigationitem-font#28795810.

[233] Stack Exchange Inc., *How to include a document into another document?*, 2017. [Online]. Available: https://tex.stackexchange.com/questions/11311/how-to-include-a-document-into-another-document.

[234] Stack Exchange Inc., *How to show subsections and subsubsections in TOC?*, 2017. [Online]. Available: https://tex.stackexchange.com/questions/17877/how-to-show-subsections-and-subsubsections-in-toc.

[235] Stack Exchange Inc., *baselinestretch vs. setspace*, 2017. [Online]. Available: https://tex.stackexchange.com/questions/79046/baselinestretch-vs-setspace.

[236] Stack Exchange Inc., *Table with footnote*, 2017. [Online]. Available: https://tex.stackexchange.com/questions/68605/table-with-footnote.

[237] Stack Exchange Inc., *How to draw a square of 1cm in LaTeX filled with color?*, 2017. [Online]. Available: https://tex.stackexchange.com/questions/106984/how-to-draw-a-square-of-1cm-in-latex-filled-with-color.

Other References

[86] Aníbal Ferreira, *Implantation of voicing on whispered speech using frequency-domain parametric modelling of source and filter information*, Proceeding of the International Symposium on Signal, Image, Video and Communications (ISIVC2016, invited paper), Nov. 2016, Tunis, Tunisia.

Appendix A

Usability Tests

A.1 Pre-Test Interview - Collected Data

Table A.1: Usability tests - Pre-test interview - Collected data

PN	Age	Gender	Occupation	UT	Test Date
1	84	Male	Retired College Professor	Beginner	September 7
2	83	Female	Retired Basic School Professor	Beginner	September 7
3	52	Female	Secretary	Beginner	September 7
4	58	Male	Factory Worker	Beginner	September 7
5	53	Female	Lawyer	Intermediate	September 7
6	55	Male	Lawyer	Intermediate	September 7
7	22	Male	College Student	Intermediate	September 7
8	52	Male	Unemployed	Beginner	September 7
9	52	Female	House Maid	Beginner	September 7
10	54	Female	House Maid	Beginner	September 8
11	27	Female	Cashier	Intermediate	September 8
12	33	Male	Sales Worker	Intermediate	September 8
13	26	Female	Software Engineer	Intermediate	September 8
14	30	Male	Catering Trainer	Intermediate	September 8
15	30	Female	Social Worker	Intermediate	September 8

PN - Participant Number

UT - User Type (Familiarization with iOS)

A.2 Tests Tasks - Collected Data - Task 1

Table A.2: Usability tests - Tests tasks - Collected data - Task 1

PN	CS	Duration	Difficulty	TN
1	Completed	120 seconds	Easy	
2	Completed	180 seconds	Easy	
3	Failed	300 seconds	Hard	
4	Completed	3 seconds	Easy	
5	Completed	3 seconds	Easy	
6	Completed	3 seconds	Easy	
7	Completed	3 seconds	Easy	
8	Completed	10 seconds	Easy	1
9	Completed	3 seconds	Easy	
10	Completed	3 seconds	Easy	
11	Completed	3 seconds	Easy	
12	Completed	3 seconds	Easy	
13	Completed	3 seconds	Easy	
14	Completed	3 seconds	Easy	
15	Completed	3 seconds	Easy	

PN - Participant Number

CS - Completion State

TN - Task Number

A.3 Tests Tasks - Collected Data - Task 2

Table A.3: Usability tests - Tests tasks - Collected data - Task 2

PN	CS	Duration	Difficulty	TN
1	Completed	180 seconds	Medium	
2	Completed	70 seconds	Easy	
3	Completed	60 seconds	Medium	
4	Completed	48 seconds	Medium	
5	Completed	4 seconds	Easy	
6	Completed	4 seconds	Easy	
7	Completed	3 seconds	Easy	
8	Completed	10 seconds	Medium	2
9	Completed	6 seconds	Easy	
10	Completed	5 seconds	Easy	
11	Completed	3 seconds	Easy	
12	Completed	4 seconds	Easy	
13	Completed	4 seconds	Easy	
14	Completed	3 seconds	Easy	
15	Completed	3 seconds	Easy	

PN - Participant Number

CS - Completion State

TN - Task Number

A.4 Tests Tasks - Collected Data - Task 3

Table A.4: Usability tests - Tests tasks - Collected data - Task 3

PN	CS	Duration	Difficulty	TN
1	Completed	120 seconds	Easy	3
2	Completed	180 seconds	Easy	
3	Completed	10 seconds	Easy	
4	Failed	210 seconds	Hard	
5	Completed	7 seconds	Easy	
6	Completed	15 seconds	Easy	
7	Completed	3 seconds	Easy	
8	Failed	423 seconds	Hard	
9	Completed	360 seconds	Hard	
10	Completed	300 seconds	Medium	
11	Completed	30 seconds	Medium	
12	Completed	26 seconds	Easy	
13	Completed	17 seconds	Easy	
14	Completed	10 seconds	Easy	
15	Completed	14 seconds	Easy	

PN - Participant Number

CS - Completion State

TN - Task Number

A.5 Tests Tasks - Collected Data - Task 4

Table A.5: Usability tests - Tests tasks - Collected data - Task 4

PN	CS	Duration	Difficulty	TN
1	Completed	22 seconds	Easy	4
2	Completed	85 seconds	Easy	
3	Completed	100 seconds	Easy	
4	Completed	28 seconds	Easy	
5	Completed	4 seconds	Easy	
6	Completed	11 seconds	Easy	
7	Completed	2 seconds	Easy	
8	Completed	60 seconds	Easy	
9	Completed	3 seconds	Easy	
10	Completed	40 seconds	Easy	
11	Completed	3 seconds	Easy	
12	Completed	5 seconds	Easy	
13	Completed	3 seconds	Easy	
14	Completed	5 seconds	Easy	
15	Completed	3 seconds	Easy	

PN - Participant Number

CS - Completion State

TN - Task Number

A.6 Tests Tasks - Collected Data - Task 5

Table A.6: Usability tests - Tests tasks - Collected data - Task 5

PN	CS	Duration	Difficulty	TN
1	Failed	300 seconds	Hard	5
2	Failed	280 seconds	Hard	
3	Completed	40 seconds	Hard	
4	Failed	100 seconds	Hard	
5	Completed	15 seconds	Easy	
6	Completed	15 seconds	Easy	
7	Completed	6 seconds	Easy	
8	Failed	420 seconds	Hard	
9	Completed	60 seconds	Easy	
10	Completed	75 seconds	Easy	
11	Completed	60 seconds	Easy	
12	Completed	35 seconds	Easy	
13	Completed	11 seconds	Easy	
14	Completed	12 seconds	Easy	
15	Completed	13 seconds	Easy	

PN - Participant Number

CS - Completion State

TN - Task Number

A.7 Participants' Opinions/Suggestions - Collected Data

Table A.7: Usability tests - Participants' opinions/suggestions - Collected data

PN	Opinions/Suggestions
1	Suggests bigger UI elements and font size, to improve readability
2	The instructions of the application should be presented at its initial launch
3	The instructions should have more specific help regarding the settings' options
4	It is not easy to understand how to exit the instructions' menu
5	The settings' icon does not provide the intended meaning; suggests a new design
6	
7	The "Pitch" slider should not appear in the *Real Voice* processing mode
8	The instructions' menu should have an easier way to exit
9	Thought the application's settings were hard to find
10	Suggested the font size should have the option of a bigger size
11	Changes in the settings should produce immediate changes
12	
13	The alerts informing about audio changes should be faster or not block UI interaction
14	
15	The instructions' menu should have a title similar to the settings' menu

PN - Participant Number

A.8 Post-Test Questionnaire - Collected Data

Table A.8: Usability tests - Post-test questionnaire - Collected data

PN	RC 1	RC 2	RC 3	RC 4	RC 5
1	6	6	6	6	6
2	6	5	6	5	6
3	5	5	5	6	6
4	5	4	4	4	4
5	5	6	5	6	6
6	6	5	5	6	6
7	6	6	5	6	6
8	6	5	5	6	6
9	6	5	5	6	6
10	6	4	5	6	5
11	6	6	5	6	6
12	6	6	5	5	6
13	5	5	5	6	5
14	6	5	5	5	6
15	6	6	6	6	6

PN - Participant Number

RC - Rating Criterion

1 - Bad

2 - Mediocre

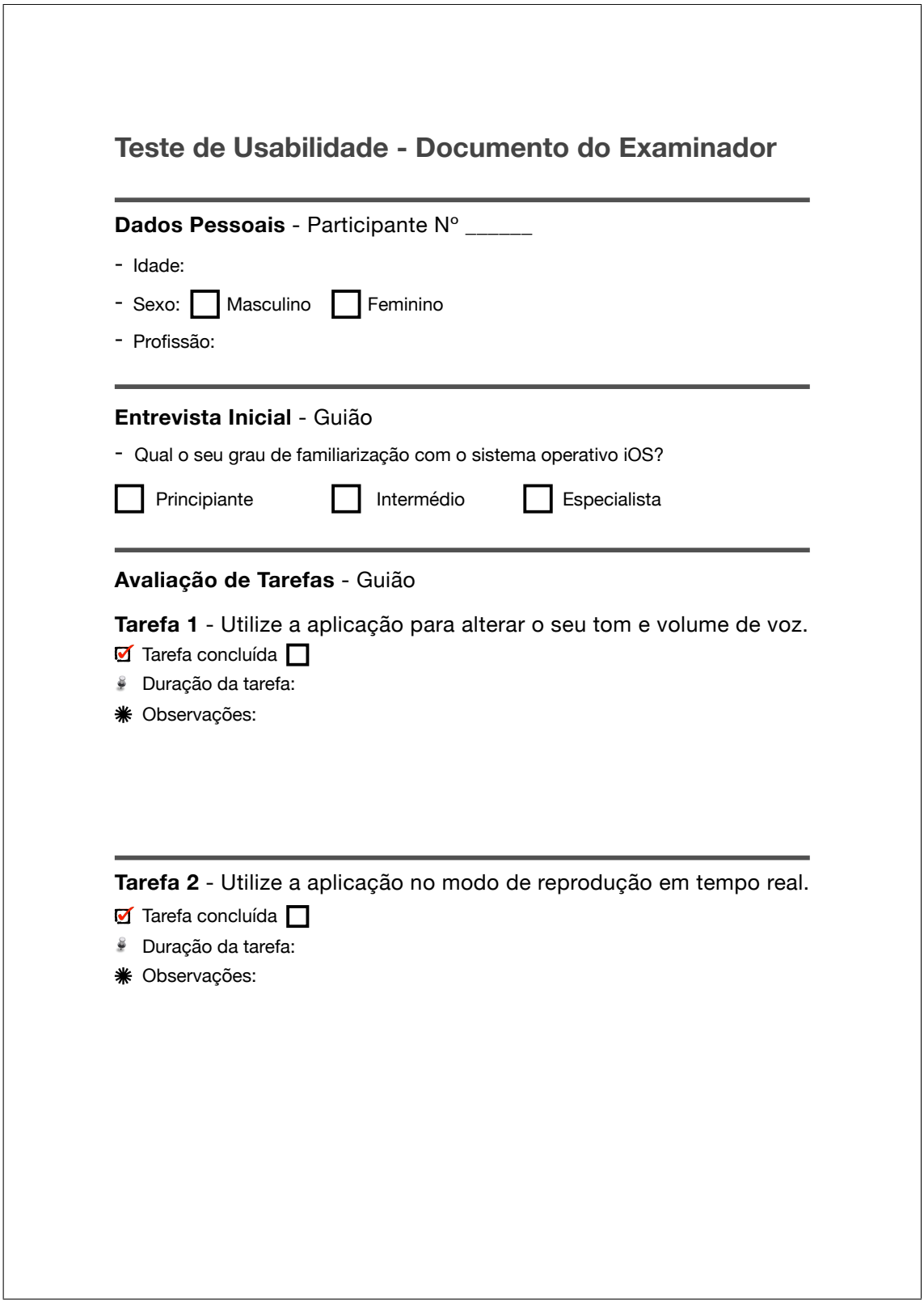
3 - Medium

4 - Good

5 - Very Good

6 - Excelent

A.9 *Facilitator Guide*



Teste de Usabilidade - Documento do Examinador

Dados Pessoais - Participante Nº _____

- Idade:
- Sexo: ☐ Masculino ☐ Feminino
- Profissão:

Entrevista Inicial - Guião

- Qual o seu grau de familiarização com o sistema operativo iOS?

☐ Principiante ☐ Intermédio ☐ Especialista

Avaliação de Tarefas - Guião

Tarefa 1 - Utilize a aplicação para alterar o seu tom e volume de voz.

☑ Tarefa concluída ☐

Duração da tarefa:

Observações:

Tarefa 2 - Utilize a aplicação no modo de reprodução em tempo real.

☑ Tarefa concluída ☐

Duração da tarefa:

Observações:

Figure A.1: Usability tests - *Facilitator Guide* - Page 1

Tarefa 3 - Altere o esquema de cores da aplicação.

☑ Tarefa concluída ☐

Duração da tarefa:

Observações:

Tarefa 4 - Encontre e visualize as instruções da aplicação.

☑ Tarefa concluída ☐

Duração da tarefa:

Observações:

Tarefa 5 - Altere o tempo máximo de gravação para 20 segundos.

☑ Tarefa concluída ☐

Duração da tarefa:

Observações:

Figure A.2: Usability tests - *Facilitator Guide* - Page 2

A.10 *Participant Guide*

Teste de Usabilidade - Tarefas - Participante Nº_____

Este conjunto de tarefas pretende servir como base ao estudo de usabilidade da aplicação *MasterVoicing*, uma aplicação, para o sistema operativo iOS, com o objectivo de ser um assistente para pessoas com afonia. A sua funcionalidade principal está simulada com a utilização de um algoritmo de mudança de tom de voz, e pretenderá utilizar futuramente um algoritmo de alteração de voz que permita a conversão de sussurros em fala audível. No fim de cada tarefa, classifique, por favor, o grau de dificuldade que sentiu na realização da tarefa, entre as opções "Fácil", "Médio" e "Difícil". Para a realização de cada tarefa, ser-lhe-hão fornecidos, para além de um dispositivo com a aplicação, um conjunto de auscultadores, com o quais poderá testar o funcionamento da mesma.

Tarefa 1 - Alterar o tom e volume da voz

- Utilize a aplicação para alterar o seu tom e volume de voz.

☑ Grau de dificuldade: ☐ Fácil ☐ Médio ☐ Difícil

Tarefa 2 - Utilizar diferentes modos de reprodução

- Utilize a aplicação no modo de reprodução em tempo real.

☑ Grau de dificuldade: ☐ Fácil ☐ Médio ☐ Difícil

Tarefa 3 - Aspecto da aplicação

- Altere o esquema de cores da aplicação.

☑ Grau de dificuldade: ☐ Fácil ☐ Médio ☐ Difícil

Tarefa 4 - Instruções

- Encontre e visualize as instruções da aplicação.

☑ Grau de dificuldade: ☐ Fácil ☐ Médio ☐ Difícil

Tarefa 5 - Limitação do tempo de gravação

- Altere o tempo máximo de gravação para 20 segundos.

☑ Grau de dificuldade: ☐ Fácil ☐ Médio ☐ Difícil

Figure A.3: Usability tests - *Participant Guide* - Page 1

Questionário Final - Avaliação Global da Aplicação

A tabela seguinte contem um conjunto de critérios relativos à aplicação. Por favor classifique cada um deles, de acordo com um valor entre **1** e **6**, que têm o seguinte significado:

- **1 -** Mau
- **2 -** Medíocre
- **3 -** Médio
- **4 -** Bom
- **5 -** Muito Bom
- **6 -** Excelente

Critérios de Avaliação	★ Classificação
- Aparência e aspectos visuais	
- Facilidade de utilização, navegação e interacção	
- Funcionalidades da aplicação	
- Desempenho e velocidade	
- Avaliação global	

Se desejar fazer alguma observação ou sugestão relativamente à avaliação da aplicação, utilize a próxima secção.

Observações/Sugestões

Muito obrigado/(a) pela sua participação!

Figure A.4: Usability tests - *Participant Guide* - Page 2

A.11 Global Rating - Calculated Values

Table A.9: Usability tests - Global rating - Calculated values

RC	Mean Value	CV	Min. Value	Max. Value
1	5,733	0,232	5,502	5,965
2	5,267	0,356	4,911	5,623
3	5,133	0,261	4,872	5,395
4	5,667	0,312	5,354	5,979
5	5,733	0,300	5,433	6,034

RC - Rating Criterion

CV - Confidence Value (95%)